



SQL Remote™
Руководство пользователя

Последнее изменение: октябрь 2002
Номер выпуска: 38133-01-0802-01

Авторские права © 1989–2002 Sybase, Inc. Неполные авторские права © 2001–2002 iAnywhere Solutions, Inc. Все права защищены.

Воспроизведение, передача или перевод настоящего документа в какой-либо форме или какими-либо средствами, электронными, механическими, ручными, оптическими или какими-либо иными может производиться только при наличии письменного разрешения со стороны компании iAnywhere Solutions, Inc. iAnywhere Solutions, Inc., является дочерней компанией Sybase, Inc.

Sybase, SYBASE (логотип), AccelaTrade, ADA Workbench, Adaptable Windowing Environment, Adaptive Component Architecture, Adaptive Server, Adaptive Server Anywhere, Adaptive Server Enterprise, Adaptive Server Enterprise Monitor, Adaptive Server Enterprise Replication, Adaptive Server Everywhere, Adaptive Server IQ, Adaptive Warehouse, AnswerBase, Anywhere Studio, Application Manager, AppModeler, APT Workbench, APT-Build, APT-Edit, APT-Execute, APT-FORMS, APT-Library, APT-Translator, ASEP, Backup Server, BayCam, Bit-Wise, BizTracker, Certified PowerBuilder Developer, Certified SYBASE Professional, Certified SYBASE Professional (логотип), ClearConnect, Client Services, Client-Library, CodeBank, Column Design, ComponentPack, Connection Manager, Convoy/DM, Copernicus, CSP, Data Pipeline, Data Workbench, DataArchitect, Database Analyzer, DataExpress, DataServer, DataWindow, DB-Library, dbQueue, Developers Workbench, Direct Connect Anywhere, DirectConnect, Distribution Director, Dynamo, e-ADK, E-Anywhere, e-Biz Integrator, E-Whatever, EC-GATEWAY, ECMAP, ECRTP, eFulfillment Accelerator, Electronic Case Management, Embedded SQL, EMS, Enterprise Application Studio, Enterprise Client/Server, Enterprise Connect, Enterprise Data Studio, Enterprise Manager, Enterprise SQL Server Manager, Enterprise Work Architecture, Enterprise Work Designer, Enterprise Work Modeler, eProcurement Accelerator, eremote, Everything Works Better When Everything Works Together, EWA, Financial Fusion, Financial Fusion Server, First Impression, Formula One, Gateway Manager, GeoPoint, iAnywhere, iAnywhere Solutions, ImpactNow, Industry Warehouse Studio, InfoMaker, Information Anywhere, Information Everywhere, InformationConnect, InstaHelp, Intelliindex, InternetBuilder, iremote, iScript, Jaguar CTS, jConnect for JDBC, KnowledgeBase, Logical Memory Manager, MainframeConnect, Maintenance Express, MAP, MDI Access Server, MDI Database Gateway, media.splash, MetaWorks, MethodSet, ML Query, MobiCATS, MySupport, Net-Gateway, Net-Library, New Era of Networks, Next Generation Learning, Next Generation Learning Studio, O DEVICE, OASIS, OASIS (логотип), ObjectConnect, ObjectCycle, OmniConnect, OmniSQL Access Module, OmniSQL Toolkit, Open Biz, Open Business Interchange, Open Client, Open Client/Server, Open Client/Server Interfaces, Open ClientConnect, Open Gateway, Open Server, Open ServerConnect, Open Solutions, Optima++, Partnerships that Work, PB-Gen, PC APT Execute, PC DB-Net, PC Net Library, PhysicalArchitect, Pocket PowerBuilder, PocketBuilder, Power Through Knowledge, Power++, power.stop, PowerAMC, PowerBuilder, PowerBuilder Foundation Class Library, PowerDesigner, PowerDimensions, PowerDynamo, Powering the New Economy, PowerJ, PowerScript, PowerSite, PowerSocket, Powersoft, Powersoft Portfolio, Powersoft Professional, PowerStage, PowerStudio, PowerTips, PowerWare Desktop, PowerWare Enterprise, ProcessAnalyst, Rapport, Relational Beans, Replication Agent, Replication Driver, Replication Server, Replication Server Manager, Replication Toolkit, Report Workbench, Report-Execute, Resource Manager, RW-DisplayLib, RW-Library, S Designor, S-Designor, S.W.I.F.T. Message Format Libraries, SAFE, SAFE/PRO, SDF, Secure SQL Server, Secure SQL Toolset, Security Guardian, SKILS, smart.partners, smart.parts, smart.script, SQL Advantage, SQL Anywhere, SQL Anywhere Studio, SQL Code Checker, SQL Debug, SQL Edit, SQL Edit/TPU, SQL Everywhere, SQL Modeler, SQL Remote, SQL Server, SQL Server Manager, SQL Server SNMP SubAgent, SQL Server/CFT, SQL Server/DBM, SQL SMART, SQL Station, SQL Toolset, SQLJ, Stage III Engineering, Startup.Com, STEP, SupportNow, Sybase Central, Sybase Client/Server Interfaces, Sybase Development Framework, Sybase Financial Server, Sybase Gateways, Sybase Learning Connection, Sybase MPP, Sybase SQL Desktop, Sybase SQL Lifecycle, Sybase SQL Workgroup, Sybase Synergy Program, Sybase User Workbench, Sybase Virtual Server Architecture, SybaseWare, Syber Financial, SyberAssist, SybMD, SyBooks, System 10, System 11, System XI (логотип), SystemTools, Tabular Data Stream, The Enterprise Client/Server Company, The Extensible Software Platform, The Future Is Wide Open, The Learning Connection, The Model For Client/Server Solutions, The Online Information Center, The Power of One, TradeForce, Transact-SQL, Translation Toolkit, Turning Imagination Into Reality, UltraLite, UNIBOM, Unilib, Uninull, Unisep, Unistring, URK Runtime Kit for UniCode, Viewer, Visual Components, VisualSpeller, VisualWriter, VQL, Warehouse Control Center, Warehouse Studio, Warehouse WORKS, WarehouseArchitect, Watcom, Watcom SQL, Watcom SQL Server, Web Deployment Kit, Web.PB, Web.SQL, WebSights, WebViewer, WorkGroup SQL Server, XA-Library, XA-Server, and XP Server являются торговыми марками компании Sybase, Inc. или ее дочерних компаний.

Все прочие торговые марки являются собственностью их соответствующих владельцев.

Последнее изменение: октябрь 2002. Номер выпуска: 38133-01-0802-01.

Содержание

	Об этом Руководстве	ix
	Документация по SQL Anywhere Studio	x
	Условные обозначения.....	xiii
	Демонстрационная база данных Adaptive Server Anywhere	xv
	Получение дополнительной информации и обратная связь	xvi
ЧАСТЬ 1		
	Введение в SQL Remote.....	1
1	Знакомство с SQL Remote	3
	Об SQL Remote	4
	Об этом Руководстве	5
2	Основные понятия SQL Remote	7
	Компоненты SQL Remote	8
	Публикации и подписка	10
	Функциональные возможности SQL Remote	12
	Типовые конфигурации системы	13
3	Установка и настройка SQL Remote.....	17
	Общие сведения по установке и настройке	18
	Подготовка сервера Adaptive Server Enterprise.....	19
	Обновление SQL Remote для Adaptive Server Enterprise.....	22
	Деинсталляция SQL Remote	23
4	Учебные разделы для пользователей Adaptive Server Anywhere.....	25
	Введение.....	26
	Учебный раздел: репликация Adaptive Server Anywhere с использованием Sybase Central	29
	Учебный раздел: репликация Adaptive Server Anywhere с использованием Interactive SQL и dbxtract	36
	Запуск репликации данных	42
	Пример публикации	45
5	Учебный раздел для пользователей Adaptive Server Enterprise	47
	Введение.....	48
	Учебный раздел: репликация Adaptive Server Enterprise	50
	Запуск репликации данных	57
ЧАСТЬ 2		
	Создание схем репликации SQL Remote	61

6	Принципы настройки SQL Remote	63
	Общие сведения о настройке.....	64
	Как реплицируются операторы	67
	Как реплицируются типы данных.....	71
	Кто и что получает?.....	73
	Ошибки и конфликты репликации.....	75
7	Настройка SQL Remote для Adaptive Server Anywhere	77
	Общие сведения о настройке.....	78
	Публикация данных.....	79
	Проектирование публикаций для Adaptive Server Anywhere	87
	Разделение таблиц, не содержащих выражение подписки	90
	Совместное использование строк в нескольких подписках	96
	Разрешение конфликтов.....	102
	Обеспечение уникальности первичных ключей	110
	Создание подписок.....	118
8	Настройка SQL Remote для Adaptive Server Enterprise	121
	Общие сведения о настройке.....	122
	Создание публикаций	123
	Проектирование публикаций для Adaptive Server Enterprise	127
	Разделение таблиц, не содержащих столбца подписки.....	129
	Совместное использование строк в нескольких подписках	136
	Управление конфликтами.....	143
	Обеспечение уникальности первичных ключей	151
	Создание подписок.....	156
ЧАСТЬ 3		
	Администрирование SQL Remote	157
9	Развертывание и синхронизация баз данных.....	159
	Общие сведения о развертывании системы	160
	Тестирование перед развертыванием системы.....	161
	Синхронизация баз данных	163
	Использование утилиты извлечения	165
	Синхронизация данных по системе передачи сообщений	172
10	Администрирование SQL Remote	173
	Обзор принципов управления	174
	Управление полномочиями SQL Remote	175
	Управление типами сообщений	183
	Работа с Message Agent	193
	Повышение производительности Message Agent	197
	Кодирование и сжатие сообщений	203
	Система отслеживания сообщений.....	205
11	Администрирование SQL Remote для Adaptive Server Anywhere.....	209
	Работа с Message Agent	210
	Сообщения об ошибках и их обработка	212
	Управление журналом транзакций и резервным копированием	216
	Использование режима ретрансляции.....	225
12	Администрирование SQL Remote для Adaptive Server Enterprise	229

	Принципы работы Message Agent для Adaptive Server Enterprise	230
	Работа с Message Agent	234
	Сообщения об ошибках и их обработка	236
	Управление журналом транзакций и резервным копированием Adaptive Server Enterprise	237
	Внесение изменений в схему	240
	Использование режима ретрансляции	241
13	Использование SQL Remote с Replication Server	243
	Когда необходимо использовать SQL Remote Open Server	244
	Архитектура систем Replication Server/SQL Remote	245
	Установка и настройка SQL Remote Open Server	248
	Конфигурирование Replication Server	250
	Разное	252
 ЧАСТЬ 4		
	Справочная информация	253
14	Справочник по утилитам и параметрам	255
	Message Agent	256
	Утилита извлечения базы данных	264
	SQL Remote Open Server	270
	Параметры SQL Remote	272
	Процедуры перехватчиков событий SQL Remote	276
15	Системные объекты для Adaptive Server Anywhere	279
	Системные таблицы SQL Remote	280
	Системные представления SQL Remote	285
16	Системные объекты для Adaptive Server Enterprise	289
	Системные таблицы SQL Remote	290
	Системные представления SQL Remote	296
	Таблицы очереди с сохранением	300
17	Справочник по командам Adaptive Server Anywhere	303
	Оператор ALTER REMOTE MESSAGE TYPE	304
	Оператор CREATE PUBLICATION	305
	Оператор CREATE REMOTE MESSAGE TYPE	306
	Оператор CREATE SUBSCRIPTION	307
	Оператор CREATE TRIGGER	308
	Оператор DROP PUBLICATION	310
	Оператор DROP REMOTE MESSAGE TYPE	311
	Оператор DROP SUBSCRIPTION	312
	Оператор GRANT CONSOLIDATE	313
	Оператор GRANT PUBLISH	314
	Оператор GRANT REMOTE	315
	Оператор GRANT REMOTE DBA	316
	Оператор PASSTHROUGH	317
	Оператор REMOTE RESET	318
	Оператор REVOKE CONSOLIDATE	319
	Оператор REVOKE PUBLISH	320
	Оператор REVOKE REMOTE	321
	Оператор REVOKE REMOTE DBA	322
	Оператор SET REMOTE OPTION	323
	Оператор START SUBSCRIPTION	324
	Оператор STOP SUBSCRIPTION	325

Оператор SYNCHRONIZE SUBSCRIPTION.....	326
Оператор UPDATE	327

18	Справочник по командам для Adaptive Server Enterprise	329
	Процедура sp_add_article	331
	Процедура sp_add_article_col.....	333
	Процедура sp_add_remote_table.....	334
	Процедура sp_create_publication.....	336
	Процедура sp_drop_publication	337
	Процедура sp_drop_remote_type.....	338
	Процедура sp_drop_sql_remote.....	339
	Процедура sp_grant_consolidate	340
	Процедура sp_grant_remote	342
	Процедура sp_link_option.....	344
	Процедура sp_modify_article.....	346
	Процедура sp_modify_remote_table	348
	Процедура sp_passthrough	349
	Процедура sp_passthrough_piece	350
	Процедура sp_passthrough_stop	352
	Процедура sp_passthrough_subscription.....	353
	Процедура sp_passthrough_user	354
	Процедура sp_populate_sql_anywhere.....	355
	Процедура sp_publisher	356
	Процедура sp_queue_clean	357
	Процедура sp_queue_confirmed_delete_old	358
	Процедура sp_queue_confirmed_transaction	359
	Процедура sp_queue_delete_old	360
	Процедура sp_queue_drop.....	361
	Процедура sp_queue_dump_database.....	362
	Процедура sp_queue_dump_transaction	363
	Процедура sp_queue_get_state	364
	Процедура sp_queue_log_transfer_reset.....	365
	Процедура sp_queue_read.....	366
	Процедура sp_queue_reset.....	367
	Процедура sp_queue_set_confirm	368
	Процедура sp_queue_set_progress	369
	Процедура sp_queue_transaction	370
	Процедура sp_remote.....	371
	Процедура sp_remote_option.....	372
	Процедура sp_remote_type.....	373
	Процедура sp_remove_article	374
	Процедура sp_remove_article_col.....	375
	Процедура sp_remove_remote_table.....	376
	Процедура sp_revoke_consolidate.....	377
	Процедура sp_revoke_remote.....	378
	Процедура sp_subscription.....	379
	Процедура sp_subscription_reset.....	380

ЧАСТЬ 5

Приложение	381
-------------------------	------------

19	Различия между SQL Remote для Adaptive Server Enterprise и SQL Remote для Adaptive Server Anywhere	383
	Типы различий	384
	Различия в функциональных возможностях.....	385
	Различия в подходах.....	386
	Ограничения при репликации Enterprise-Enterprise	388

20	Поддерживаемые платформы и системы передачи сообщений.....	389
	Поддерживаемые системы передачи сообщений	390
	Поддерживаемые операционные системы.....	391
21	Индекс	393

Об этом Руководстве

О чем эта книга?

В этой книге рассматриваются функциональные возможности и особенности системы репликации данных для мобильных компьютеров SQL Remote. Эта система позволяет обеспечить совместный доступ к данным из отдельной базы данных Adaptive Server Anywhere или Adaptive Server Enterprise и множества баз данных Adaptive Server Anywhere посредством обмена информацией через непрямую систему связи, например, электронную почту или сервисы передачи файлов.

Для кого предназначена эта книга?

Это Руководство предназначено для пользователей Adaptive Server Anywhere и Adaptive Server Enterprise, желающих добавить к имеющимся системам управления информацией систему репликации SQL Remote.

Перед началом работы

☞ Для сравнения SQL Remote с другими технологиями репликации данных см. раздел "Технологии репликации" (Replication Technologies) на стр. 19 в документе *"Введение в SQL Anywhere Studio" (Introducing SQL Anywhere Studio)*.

Документация по SQL Anywhere Studio

Этот документ входит в состав документации по семейству систем SQL Anywhere. В данном разделе приведен список других книг в составе этой документации с их кратким описанием, а также рекомендации по работе с ними.

Комплект документов по SQL Anywhere Studio

Комплект документов по SQL Anywhere Studio включает в себя следующие книги:

- ◆ **Введение в SQL Anywhere Studio (Introducing SQL Anywhere Studio).** В этом документе содержится обзор процесса управления базами данных SQL Anywhere Studio, а также технологий синхронизации. В него также включены учебные разделы, предназначенные для ознакомления читателей с каждым из компонентов SQL Anywhere Studio.
- ◆ **Новое в SQL Anywhere Studio (What's New in SQL Anywhere Studio).** Этот документ предназначен для пользователей, работавших с предыдущими версиями данного ПО. В нем приводится список новых функций в сравнении с прошлыми выпусками данного продукта и описание процедур обновления.
- ◆ **Введение в Adaptive Server Anywhere (Adaptive Server Anywhere Getting Started).** Этот документ предназначен для пользователей, ранее не работавших с реляционными базами данных или не знакомых с Adaptive Server Anywhere. Он содержит краткое руководство, позволяющее немедленно начать использование СУБД Adaptive Server Anywhere, а также вводный материал по проектированию и разработке баз данных и работе с ними.
- ◆ **Руководство по администрированию баз данных Adaptive Server Anywhere (Adaptive Server Anywhere Database Administration Guide).** В этом документе представлен обширный набор сведений о работе с базами данных, управлении ими и их конфигурированию.
- ◆ **Руководство пользователя SQL Adaptive Server Anywhere (Adaptive Server Anywhere SQL User's Guide).** В этом документе описывается процесс проектирования и создания баз данных, а также такие действия, как импортирование, экспортирование и изменение данных, выполнение запросов и формирование хранимых процедур и триггеров.
- ◆ **Справочник по SQL для Adaptive Server Anywhere (Adaptive Server Anywhere SQL Reference Manual).** В этом документе представлено полное справочное руководство по языку SQL, используемому системой Adaptive Server Anywhere. В нем также содержится описание системных таблиц и процедур Adaptive Server Anywhere.
- ◆ **Руководство по программированию Adaptive Server Anywhere (Adaptive Server Anywhere Programming Guide).** Этот документ содержит руководство по разработке и развертыванию приложений баз данных с использованием языков программирования C, C++ и Java. Пользователи таких средств разработки, как Visual Basic и PowerBuilder, могут использовать интерфейсы программирования, предоставляемые этими средствами.
- ◆ **Сообщения об ошибках Adaptive Server Anywhere (Adaptive Server Anywhere Error Messages).** В этом документе представлен полный перечень сообщений об ошибках системы Adaptive Server Anywhere вместе с информацией по диагностике.
- ◆ **Защита C2 в Adaptive Server Anywhere (Adaptive Server Anywhere C2 Security Security).** Правительство США присвоило системе Adaptive Server

Anywhere 7.0 определенный уровень защиты C2 по классификации TCSEC (Trusted Computer System Evaluation Criteria, Критерии анализа надежности компьютерных систем). Этот документ может представлять интерес для тех, кто хочет использовать данную версию Adaptive Server Anywhere в стиле приложений среды уровня защиты C2. В этом документе *не* содержится описание функций безопасности, встроенных в продукт после сертификации.

- ◆ **Руководство пользователя по системе синхронизации MobiLink (MobiLink Synchronization User's Guide).** В этом документе описываются особенности системы синхронизации данных MobiLink для мобильных вычислений, реализующей совместную работу с данными отдельной БД Oracle, Sybase, Microsoft или IBM и множества БД Adaptive Server Anywhere или UltraLite.
- ◆ **Руководство пользователя SQL Remote (SQL Remote User's Guide).** В этой книге описываются особенности системы репликации данных для мобильных компьютеров SQL Remote, реализующей совместную работу с данными отдельной БД Adaptive Server Anywhere или Adaptive Server Enterprise и множества БД Adaptive Server Anywhere через непрямую систему связи, например, электронную почту или сервисы передачи файлов.
- ◆ **Руководство пользователя по UltraLite (UltraLite User's Guide).** В этой книге описывается процесс разработки приложений баз данных для небольших устройств, например, карманных органайзеров, с использованием технологии развертывания UltraLite для баз данных Adaptive Server Anywhere.
- ◆ **Руководство пользователя по UltraLite для PenRight! MobileBuilder (UltraLite User's Guide for PenRight! MobileBuilder).** Эта книга предназначена для пользователей средства разработки PenRight! MobileBuilder. В нем описывается, как использовать технологию UltraLite в среде программирования MobileBuilder.
- ◆ **Справка про SQL Anywhere Studio (SQL Anywhere Studio Help).** Этот документ доступен только в режиме online. Он содержит контекстно-зависимую справку для приложений Sybase Central, Interactive SQL и других графических средств.

Кроме этого комплекта документов, приложения SQL Modeler и InfoMaker имеют собственную online-документацию.

Форматы документации

Документация по SQL Anywhere Studio доступна в следующих форматах:

- ◆ **Online-книги.** В виде online-книг предоставляется полная документация по SQL Anywhere Studio, включая документы, имеющиеся в печатном виде, и контекстно-зависимую справку по средствам SQL Anywhere. Содержание online-книг обновляется с выходом каждой обновленной версии продукта; они являются самым полным и актуальным источником информации о продуктах на любой момент времени.

Для обращения к online-книгам в операционных системах Windows выберите Start (Пуск) ► Programs (Программы) ► Sybase SQL Anywhere 8 ► Online Books. Для выполнения навигации по online-книгам можно использовать оглавление HTML-справки, индекс, панель поиска в левой области окна, а также ссылки и меню в правой области окна.

Для обращения к online-книгам в операционных системах UNIX введите в командной строке и выполните следующую команду:

```
dbbooks
```

- ◆ **Книги для печати.** Документация по SQL Anywhere представлена также в виде набора файлов формата PDF, просматриваемых в Adobe Acrobat Reader. PDF-файлы можно найти на CD-ROM в каталоге *pdf_docs*. Инсталлировать их можно в процессе работы программы установки путем выбора соответствующих пунктов.
- ◆ **Печатные книги.** В комплект поставки SQL Anywhere Studio включены следующие документы:
 - ◆ *Введение в SQL Anywhere Studio (Introducing SQL Anywhere Studio)*;
 - ◆ *Введение в Adaptive Server Anywhere (Adaptive Server Anywhere Getting Started)*;
 - ◆ *Краткий справочник по SQL Anywhere Studio (SQL Anywhere Studio Quick Reference)*. Перечисленные документы доступны только в печатной форме.

Полный набор документов доступен в виде комплекта документов по SQL Anywhere в отделе сбыта Sybase или в online-магазине Sybase по адресу <http://e-shop.sybase.com/cgi-bin/eshop.storefront/>.

Условные обозначения

В данном разделе перечислены символьные и графические условные обозначения, используемые в этой документации.

Условные обозначения синтаксиса

В описаниях синтаксиса SQL используются следующие условные обозначения:

- ◆ **Ключевые слова.** Все ключевые слова SQL показаны так же, как слова ALTER TABLE в следующем примере:

ALTER TABLE [*владелец*.]*имя-таблицы*

- ◆ **Метки-заполнители.** Элементы, которые должны быть заменены соответствующими идентификаторами или выражениями, показаны так же, как слова *владелец* и *имя-таблицы* в следующем примере.

ALTER TABLE [*владелец*.]*имя-таблицы*

- ◆ **Повторяющиеся элементы.** Список повторяющихся элементов обозначается элементом этого списка с многоточием после него, как выражение *ограничение-столбца* в следующем примере:

ADD определение-столбца [*ограничение-столбца*, ...]

Конструкция может содержать один и более элементов списка. Если элементов несколько, они должны быть разделены запятыми.

- ◆ **Необязательные элементы.** Необязательные элементы оператора заключаются в квадратные скобки.

RELEASE SAVEPOINT [*имя-точки-сохранения*]

Здесь квадратные скобки означают, что указывать *имя-точки-сохранения* не обязательно. При написании кода квадратные скобки вводить не следует.

- ◆ **Параметры.** Если из списка можно выбрать только один элемент (или не выбрать вообще), то эти элементы отделяются вертикальной чертой, а сам список заключается в квадратные скобки.

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

В этом примере можно выбрать элемент ASC, элемент DESC или не выбрать ни один из них. При написании кода квадратные скобки вводить не следует.

- ◆ **Взаимоисключающие варианты.** Если можно выбрать только один элемент из списка (и выбор должен быть сделан), то такие взаимоисключающие варианты заключаются в фигурные скобки.

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

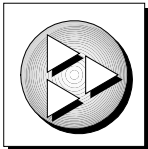
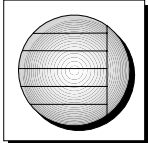
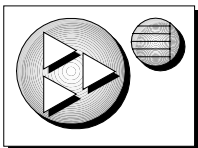
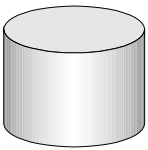

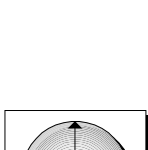

Если выбран параметр QUOTES, необходимо указать один из вариантов ON или OFF. При написании кода квадратные и фигурные скобки вводить не следует.

- ◆ **Один или более параметров.** При выборе более одного параметра следует отделить их друг от друга запятыми.

{ **CONNECT**, **DBA**, **RESOURCE** }

Графические значки

В этом документе используются следующие значки:

Значок	Значение
	<p>Клиентское приложение.</p>
	<p>Сервер базы данных, например, Sybase Adaptive Server Anywhere или Adaptive Server Enterprise.</p>
	<p>Приложение и сервер базы данных UltraLite. В системе UltraLite сервер базы данных и приложение являются частью одного и того же процесса.</p>
	<p>База данных. В некоторых сложных диаграммах значок может использоваться для обозначения как самой базы данных, так и сервера базы данных, под управлением которого она находится.</p>
	<p>Микропрограммные средства репликации или синхронизации. Микропрограммные средства являются вспомогательными средствами для обеспечения совместного использования данных различными БД. Это, например, MobiLink Synchronization Server, SQL Remote Message Agent и Replication Agent (Log Transfer Manager), используемые совместно с Replication Server.</p>
	<p>Сервер Sybase Replication Server.</p>
	<p>Интерфейс программирования.</p>

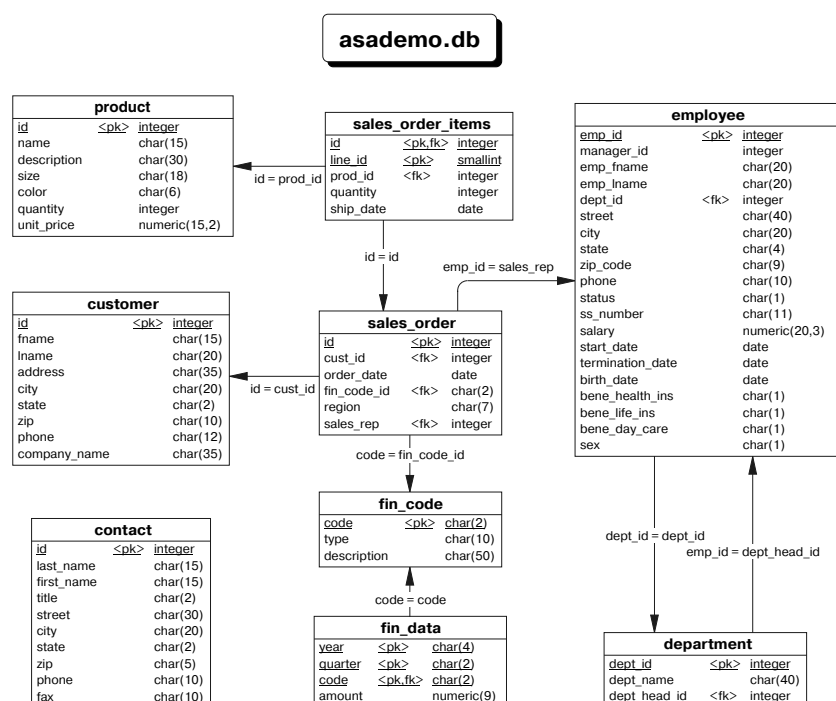
Демонстрационная база данных Adaptive Server Anywhere

Многие из примеров в данном документе относятся к демонстрационной базе данных Adaptive Server Anywhere.

Демонстрационная база данных содержится в файле *asademo.db*, находящемся в папке, где установлена система SQL Anywhere.

Демонстрационная база данных представляет собой базу данных небольшой компании. Она содержит внутреннюю информацию о компании (служащие, отделы, финансовые данные), а также информацию о продукции (продукты) и сбыте (заказы на покупку, клиенты, контакты). Вся информация в базе данных является вымышленной.

На следующем рисунке приведены таблицы в демонстрационной базе данных и их взаимосвязи.



Получение дополнительной информации и обратная связь

Нам было бы интересно узнать мнение читателей об этой документации, а также получить предложения и отзывы.

Отправить свой отзыв на эту документацию и программное обеспечение можно с помощью групп новостей, специально предназначенных для обсуждения технологий SQL Anywhere. Эти группы новостей можно найти на сервере групп новостей *forums.sybase.com*.

Имеются следующие группы новостей:

- ◆ `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`

Заявление по группам новостей

Компания iAnywhere Solutions не обязана предоставлять какие-либо решения, информацию или идеи относительно своих групп новостей, а также не обязана обеспечивать наличие обслуживающего персонала, кроме системного оператора, контролирующего процесс предоставления обслуживания и обеспечивающего его функционирование и доступность.

Технические консультанты и другой персонал компании iAnywhere Solutions занимаются поддержкой групп новостей при наличии свободного времени. Они предлагают свою помощь на добровольной основе и не занимаются постоянной деятельностью по решению проблем и предоставлению информации. Их возможности по оказанию помощи зависят от рабочей нагрузки.

Введение в SQL Remote

В этой части представлены основные понятия, архитектура и функции системы SQL Remote.

Приведенная в данной части информация относится как к SQL Remote для Adaptive Server Anywhere, так и к SQL Remote для Adaptive Server Enterprise.



Г Л А В А 1

Знакомство с SQL Remote

Об этой главе В данной главе приводится общее описание системы SQL Remote и соответствующей документации.

Содержание

Раздел	Страница
Об SQL Remote	4
Об этом Руководстве	5

Об SQL Remote

SQL Remote - это технология репликации данных, позволяющая выполнять двунаправленную репликацию данных между центральным сервером данных и большим количеством удаленных баз данных, в число которых также, как правило, входит множество мобильных баз данных.

Технология репликации SQL Remote использует принцип обмена сообщениями и не требует наличия прямого подключения серверов. Передача данных может осуществляться посредством периодического установления коммутируемых соединений или с помощью электронной почты.

Требования по ресурсам и администрированию на удаленных узлах минимальны. Задержку по времени при передаче данных между консолидированной и удаленной базами данных можно изменить в пределах от нескольких минут до нескольких часов или дней.

Технология Sybase SQL Remote реализуется в следующих двух формах:

- ◆ **SQL Remote для Adaptive Server Anywhere.** Обеспечивает репликацию данных между консолидированной БД Adaptive Server Anywhere и большим количеством удаленных БД.
- ◆ **SQL Remote для Adaptive Server Enterprise.** Обеспечивает репликацию данных между консолидированной БД Adaptive Server Enterprise и большим количеством удаленных БД Adaptive Server Anywhere.

В настоящем документе рассмотрены обе технологии.

Необходимо наличие надлежащим образом лицензированного программного обеспечения SQL Remote для каждой базы данных, входящей в установку SQL Remote.

☞ Для получения подробной информации об основных понятиях и функциях SQL Remote см. раздел "Основные понятия SQL Remote" на стр. 7.

☞ Список поддерживаемых операционных систем и систем передачи сообщений см. в разделе "Поддерживаемые платформы и системы передачи сообщений" на стр. 389.

Об этом Руководстве

В данном Руководстве рассматриваются вопросы, связанные с проектированием, построением и обслуживанием систем репликации SQL Remote.

Руководство включает следующие части:

- ◆ **Введение в SQL Remote.** Основные понятия и средства репликации SQL Remote.
- ◆ **Создание схем репликации для SQL Remote.** Разработка схем репликации для применения в инсталляции SQL Remote.
- ◆ **Администрирование SQL Remote.** Развертывание баз данных SQL Remote и администрирование инсталляции SQL Remote.
- ◆ **Справочная информация.** Команды, системные таблицы и другой справочный материал по SQL Remote.

Инсталляция продукта

В данном разделе описывается процедура установки SQL Remote для Adaptive Server Enterprise. Если ПО SQL Remote было приобретено в составе других программных пакетов, обратитесь к инструкциям по инсталляции этих программных пакетов.

- ❖ **Процедура установки программного обеспечения SQL Remote (Windows)**
 - 1 Вставьте установочный компакт-диск в привод CD-ROM.
 - 2 Если программа установки не запускается автоматически, запустите приложение *setup* с компакт-диска.
 - 3 Выполняйте указания программы установки.
- ❖ **Процедура установки программного обеспечения SQL Remote (UNIX)**
 - ◆ Следуйте соответствующим указаниям для конкретной операционной системы, приведенным в документе *"Подготовка к работе с Adaptive Server Anywhere"* (*Adaptive Server Anywhere Read Me First*).

☞ При использовании SQL Remote для Adaptive Server Enterprise необходимо установить SQL Remote в любую базу данных, которая будет задействована при репликации. Для получения информации о добавлении функций SQL Remote в базу данных см. раздел "Установка и настройка SQL Remote" на стр. 19.

Г Л А В А 2

Основные понятия SQL Remote

Об этой главе

В этой главе рассматриваются основные понятия, принципы и функции системы репликации SQL Remote.

Содержание

Раздел	Страница
Компоненты SQL Remote	8
Публикации и подписка	10
Функциональные возможности SQL Remote	12
Типовые конфигурации системы	13

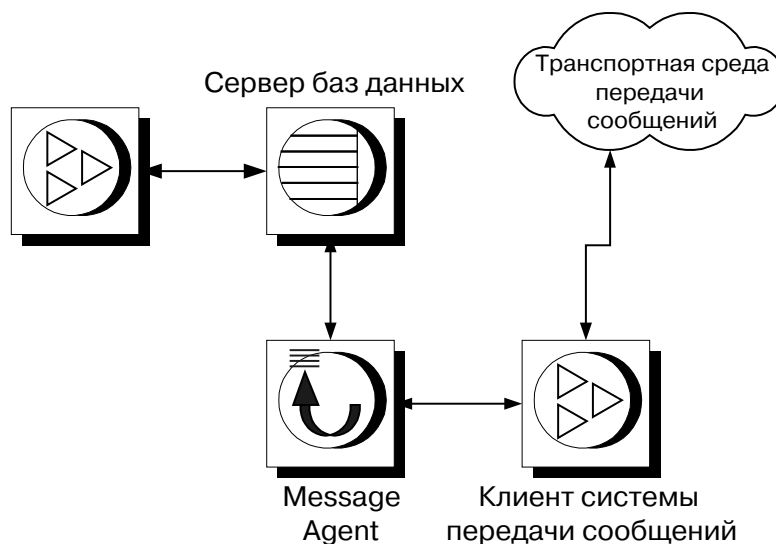
Компоненты SQL Remote

При использовании системы SQL Remote требуется наличие следующих компонентов:

- ◆ **Сервер баз данных.** Для обработки данных на каждом узле необходимо наличие СУБД Adaptive Server Anywhere или Adaptive Server Enterprise.
- ◆ **Message Agent.** Утилита SQL Remote Message Agent требуется на консолидированном узле и на каждом удаленном узле для отправки и получения сообщений SQL Remote.

Message Agent подключается к серверу данных при помощи подключения клиент-сервер. Это может выполняться на машине, которая является сервером данных, или на другой машине.

- ◆ **Утилита извлечения базы данных.** Утилита извлечения используется для подготовки удаленных БД для репликации с консолидированной БД в период разработки и тестирования, а также при развертывании системы.
- ◆ **Клиентское программное обеспечение системы передачи сообщений.** Для осуществления обмена сообщениями при репликации SQL Remote использует существующие системы передачи сообщений. Также предусмотрена система передачи сообщений путем совместного доступа к файлам, не требующая специального клиентского программного обеспечения. На каждом компьютере, задействованном в репликации SQL Remote и использующем иную систему передачи сообщений, нежели система передачи сообщений путем совместного доступа к файлам, должна быть установлено ПО для данной системы передачи сообщений.
- ◆ **Клиентские приложения.** Приложения для работы с базами данных SQL Remote, являются стандартными приложениями БД типа "клиент-сервер".



Сервер баз данных

В качестве сервера данных может использоваться как сервер Adaptive Server Enterprise, так и сервер Adaptive Server Anywhere. Сервер баз данных на удаленном узле, как правило, представляет собой персональный сервер с Adaptive Server Anywhere, но им может также быть сервер Adaptive Server Enterprise или сервер Adaptive Server Anywhere.

Клиентские приложения

Клиентские приложения работают с данными в базе данных. Клиентские приложения используют один из интерфейсов "клиент-сервер", поддерживаемых сервером данных:

- ◆ Клиентское приложение для Adaptive Server Anywhere может использовать ODBC, Embedded SQL или Sybase Open Client.
- ◆ Клиентское приложение для Adaptive Server Enterprise может использовать один из интерфейсов Sybase Client Server, ODBC или Embedded SQL.

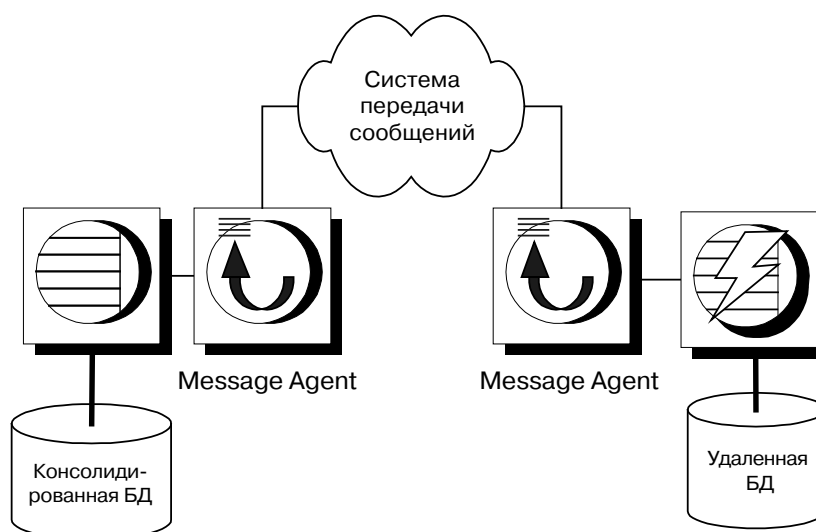
Работа клиентских приложений не зависит от того, находится это приложение на стороне консолидированной или удаленной базы данных. С точки зрения клиентского приложения разницы между этими ситуациями нет.

Message Agent

SQL Remote **Message Agent** осуществляет отправку и получение репликационных сообщений. Message Agent – это клиентское приложение, которое управляет обменом сообщениями между базами данных. Message Agent должен быть установлен как в консолидированных, так и в удаленных узлах.

Message Agent для Adaptive Server Anywhere представляет собой программу с именем *dbremote.exe* в операционных системах PC или *dbremote* в UNIX.

Message Agent для Adaptive Server Enterprise представляет собой программу с именем *ssremote.exe* в операционных системах PC или *ssremote* в UNIX.



Клиентское ПО системы передачи сообщений

При использовании системы передачи сообщений путем совместного доступа к файлам клиентское программное обеспечение системы передачи сообщений не требуется.

При использовании электронной почты или другой системы обмена сообщениями для отправки и получения сообщений необходимо наличие клиентского ПО для этой системы.

Публикации и подписка

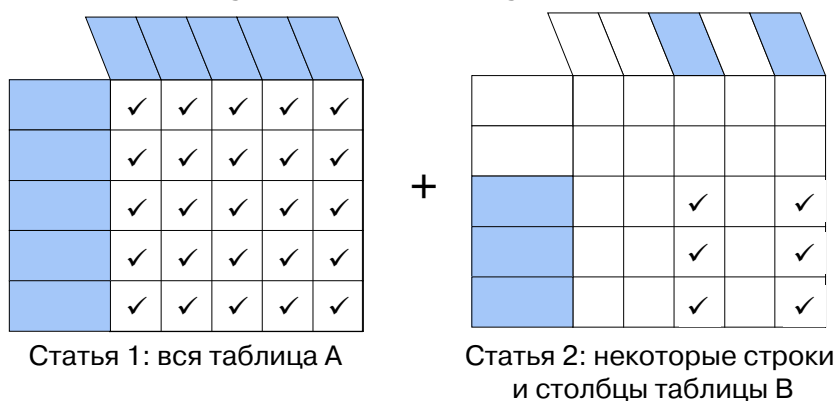
Данные, реплицируемые в SQL Remote, объединяются в **публикации**. Каждая база данных, которая использует информацию в публикации совместно с другой базой данных, должна иметь **подписку** на публикацию.

Данные организованы в публикации

Публикация – это объект базы данных, содержащий описание подлежащих репликации данных. Удаленные пользователи какой-либо базы данных, которым необходимо получать указанные в публикации данные, должны **подписаться** на эту публикацию.

Публикация может включать данные из нескольких таблиц базы данных. Данные из таблицы, представленной в публикации, называется **статьей**. Каждая статья может состоять как из целой таблицы, так и из поднабора строк и столбцов таблицы.

Публикация из двух таблиц



Произведенные изменения каждой публикации в базе данных периодически реплицируются всем подписчикам на данную публикацию. Такая репликация называется **обновлением** публикации.

Двунаправленная передача сообщений

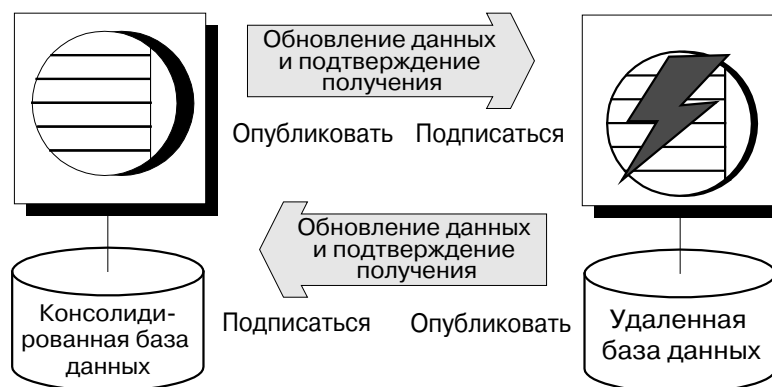
Удаленные базы данных подписываются на публикации консолидированной базы данных, что позволяет им получать данные из консолидированной БД. Для этого в консолидированной базе данных создается **подписка**, в которой подписчики определяются по имени и публикациям, которые они будут получать.

Передача сообщений в SQL Remote всегда осуществляется двунаправлено. Консолидированная база данных посылает в удаленные БД сообщения, содержащие обновления публикаций, и удаленные БД также посылают сообщения в консолидированную БД.

Например, если данные в публикации в консолидированной базе данных обновлены, то эти обновления направляются удаленным базам данных. Даже если данные в удаленной БД никогда не обновляются, в консолидированную базу данных должны направляться **сообщения с подтверждением** в целях контроля за состоянием процесса репликации.

Подписываются обе базы данных

Сообщения должны направляться в обе стороны, поэтому подписку на публикацию (созданную в консолидированной базе данных) выполняет не только удаленная база данных, но также и консолидированная база данных, которая должна подписаться на соответствующую публикацию, созданную в удаленной базе данных.



Синхронизация удаленной базы данных

Когда пользователи удаленной базы данных изменяют свои собственные копии данных, их изменения реплицируются в консолидированную базу данных. После того как сообщения, содержащие изменения, обработаны в консолидированной БД, изменения становятся частью публикации консолидированной БД и включаются в следующий раунд обновлений всех удаленных БД (кроме БД-источника изменений). Таким образом, репликация от удаленного узла в удаленный узел осуществляется через консолидированную базу данных.

Когда подписка первоначально настроена, а обе базы данных приведены в состояние, в котором они содержат один и тот же набор информации, все готово к началу репликации. Процесс настройки удаленной базы данных с целью достижения согласованности с консолидированной базой данных называется **синхронизацией**. Синхронизация может выполняться как вручную, так и автоматически с помощью утилиты извлечения базы данных. Утилита извлечения может выполняться либо как утилита командной строки, либо, при использовании для консолидированной БД Adaptive Server Anywhere, из Sybase Central.

При использовании для создания удаленной БД утилиты извлечения базы данных SQL Remote соответствующие публикация и подписка автоматически создаются в удаленных базах данных.

Функциональные возможности SQL Remote

SQL Remote обладает следующими наиболее важными функциональными возможностями.

Поддержка большого числа подписчиков. SQL Remote предназначен для поддержки репликации с большим количеством подписчиков на публикацию.

Эта возможность особенно важна при наличии интенсивно работающих мобильных приложений, которым может потребоваться репликация на портативные компьютеры данных сотен или тысяч торговых представителей из единственной офисной базы данных.

Репликация на основе журнала транзакций. Репликация SQL Remote основана на журнале транзакций. Это позволяет при каждом обновлении производить репликацию только изменений в данных, а не самих данных. Кроме того, по сравнению с другими технологиями репликация способ репликации на основе журнала имеет преимущества по производительности.

В журнале транзакций хранятся записи обо всех произведенных изменениях базы данных. SQL Remote реплицирует внесенные в базы данных изменения согласно записям в журнале транзакций. Согласно журналу транзакций консолидированной базы данных, удаленным базам данных периодически направляются все подтвержденные транзакции (относящиеся к любой публикации). В удаленных узлах все подтвержденные транзакции в журнале транзакций периодически передаются консолидированной базе данных.

Благодаря тому, что реплицируются только подтвержденные транзакции, SQL Remote обеспечивает надлежащую атомарность транзакций во всей системе репликации, а также поддерживает согласованность среди баз данных, задействованных в репликации, несмотря на наличие некоторой временной задержки при реплицировании данных.

Централизованное администрирование. В SQL Remote администрирование осуществляется централизованно в консолидированной базе данных. Это особенно важно при интенсивно работающих мобильных приложениях, где пользователям портативных компьютеров не придется выполнять задачи администрирования базы данных. Кроме того, это также важно, когда в репликации участвуют маленькие офисы, которые имеют серверы, но имеют мало административных ресурсов.

Задачи администрирования включают создание и поддержку публикаций, удаленных пользователей и подписок, а также исправление ошибок и разрешение конфликтов, если таковые имеют место.

Экономичные требования ресурсов. Единственное программное обеспечение, требуемое для выполнения SQL Remote в дополнение к СУБД Adaptive Server Anywhere или Adaptive Server Enterprise – это Message Agent и ПО системы передачи сообщений. При использовании системы передачи сообщений путем совместного доступа к файлам специальное программное обеспечение не требуется, поскольку каждый удаленный пользователь получает доступ к папке, в которой хранятся файлы сообщений.

Для всех компонентов системы репликации поддерживаются умеренные требования к объемам памяти и дискового пространства, что позволяет устанавливать SQL Remote без привлечения дополнительных средств на установку нового аппаратного обеспечения.

Поддержка различных платформ. SQL Remote поставляется для ряда операционных систем и систем передачи сообщений.

☞ Список поддерживаемых платформ см. в разделе "Поддерживаемые платформы и системы передачи сообщений" на стр. 449.

Типовые конфигурации системы

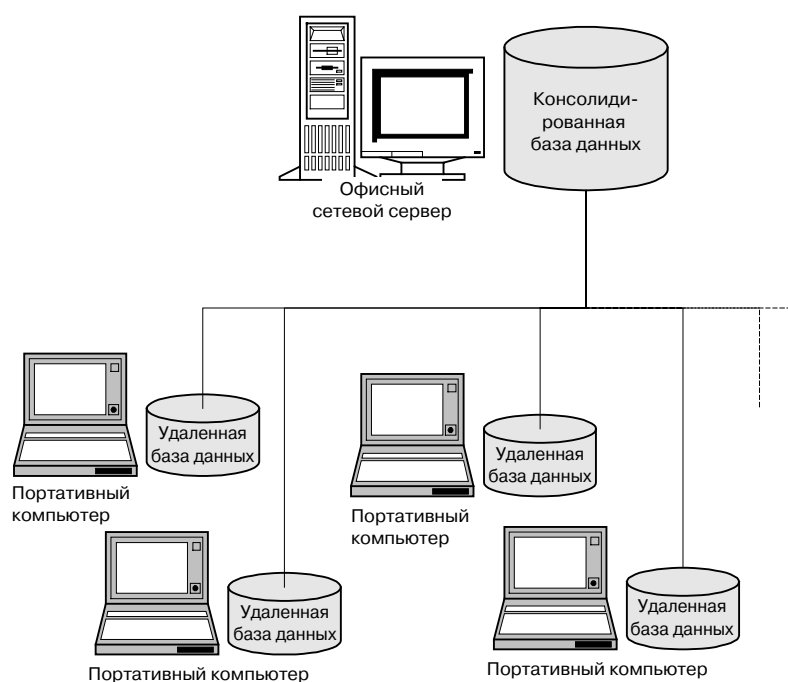
Хотя система SQL Remote может обеспечивать услуги репликации во многих различных средах, ее разработка осуществлялась исходя из следующих принципов:

- ◆ Система SQL Remote является решением, при котором нагрузка по администрированию не лежит на удаленных пользователях, как, например, в случае интенсивно работающих мобильных приложений.
- ◆ Передача данных между узлами может происходить время от времени и по непрямому каналу, наличие постоянного прямого соединения не обязательно.
- ◆ Наиболее важным моментом является соблюдение требований к памяти и ресурсам в удаленных узлах.

В следующих примерах приведены некоторые типичные установки SQL Remote.

Репликация "сервер – портативный компьютер" для интенсивно работающих мобильных приложений

SQL Remote обеспечивает двунаправленную репликацию между базой данных, расположенной в офисной сети, и персональными базами данных в портативных компьютерах торговых представителей. При такой конфигурации для передачи сообщений может использоваться электронная почта.



На главном сервере может быть установлено серверное ПО для управления базой данных этой компании. Message Agent в базе данных компании выполняется как клиентское приложение для этого сервера.

На портативном компьютере каждого торгового представителя устанавливается персональный сервер Adaptive Server Anywhere для управления собственными данными.

Находясь вне офиса, торговому представителю достаточно один раз со своего компьютера выполнить подключение по телефонной линии, чтобы выполнить следующие операции:

- ◆ Получить новую электронную почту;
- ◆ Отправить все подготовленные к отправке электронные сообщения;
- ◆ Получить с главного сервера обновления публикаций;
- ◆ Отправить на главный сервер все локальные обновления, например, информацию о новых заказах.

Обновления могут включать, например, новые специальные предложения на продукты, с которыми имеет дело торговый представитель, или новую информацию по ценам и запасам на складе. Message Agent на портативном компьютере считывает эти обновления и автоматически обрабатывает их (выполняет соответствующие операции с данными) в базе данных торгового представителя. При этом не требуется каких-либо дополнительных действий со стороны торгового представителя.

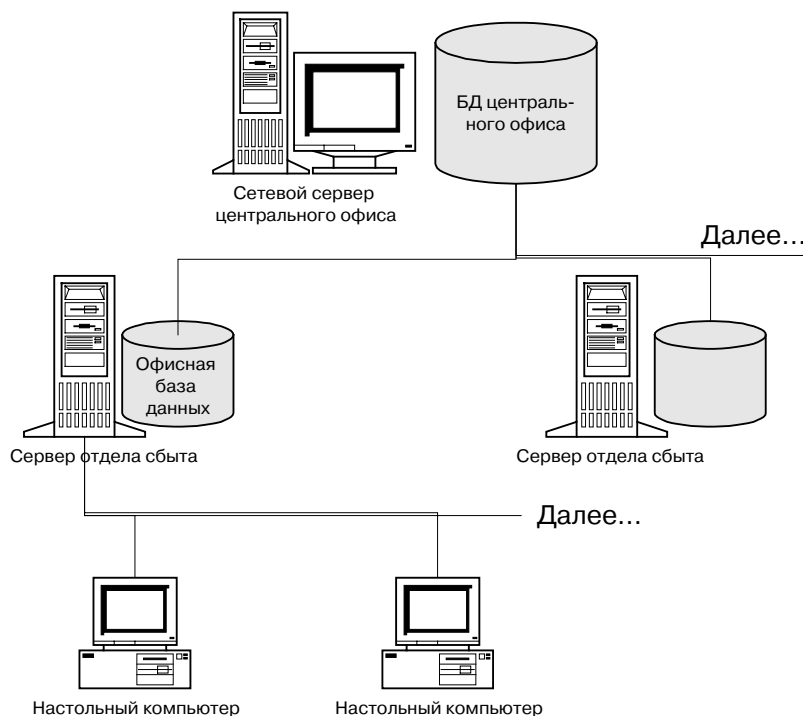
Новые заказы, введенные торговым представителем, также автоматически передаются в офис без каких-либо дополнительных действий со стороны торгового представителя.

Репликация "сервер – сервер" между офисами

SQL Remote обеспечивает двунаправленную репликацию между серверами баз данных отделов сбыта или торговых точек и центральным офисом компании, при которой не требуется выполнять администрирование в каждом офисе компании, за исключением начальной установки и настройки системы и задач по поддержанию сервера.

SQL Remote не предназначен для использования в случаях, когда на каждом узле требуется постоянная доступность новейших данных с других узлов. Вместо этого SQL Remote может использоваться тогда, когда допустимо копирование данных приблизительно с часовым интервалом.

При такой конфигурации для обмена сообщениями может использоваться электронная почта (если такая система уже имеется во всей компании). В качестве альтернативы для реализации системы передачи сообщений FILE может использоваться программное обеспечение передачи файлов и системы периодического коммутируемого доступа.



SQL Remote легко конфигурируется, что позволяет обеспечить доставку в каждый офис необходимых данных. Конфиденциальность таблиц, содержащих информацию только для внутреннего использования (например, данные о сотрудниках, если это франчайзинговая компания), может обеспечиваться даже при хранении таких таблиц в той же базе данных, что и реплицируемые данные.

К иерархиям SQL Remote можно добавлять уровни: например, каждый сервер отдела сбыта мог бы действовать как консолидированная база данных, поддерживая удаленных подписчиков, которые работают из этого офиса.

ГЛАВА 3

Установка и настройка SQL Remote

Об этой главе

В данной главе описан процесс добавления функций SQL Remote на сервер Adaptive Server Enterprise.

Только для пользователей Adaptive Server Enterprise

Эта глава предназначена только для пользователей SQL Remote для Adaptive Server Enterprise. Функции SQL Remote автоматически устанавливаются в базы данных Adaptive Server Anywhere.

Информация в данной главе изложена с учетом того, что программное обеспечение SQL Remote уже установлено на компьютере.

Содержание

Раздел	Страница
Общие сведения по установке и настройке	18
Подготовка сервера Adaptive Server Enterprise	19
Обновление SQL Remote для Adaptive Server Enterprise	22
Деинсталляция SQL Remote	23

Общие сведения по установке и настройке

В данном Руководстве набор баз данных с возможностью обмена информацией при помощи SQL Remote называется **инсталляцией SQL Remote**. С физической точки зрения система SQL Remote может состоять из сотен или даже тысяч баз данных, совместно использующих информацию; однако из-за того, что в каждой физической базе данных SQL Remote сохраняет информацию согласованной на уровне транзакций с другими физическими базами данных, всю систему можно представить как одну **рассредоточенную базу данных**.

При создании крупной системы SQL Remote базы данных могут быть установлены на большом количестве компьютеров. Поскольку изменения в конфигурацию и схемы репликации могут вноситься уже в установленной системе, настоятельно рекомендуется приступать к выполнению этих действий только после тщательного анализа и тестирования проекта.

Задачи настройки

При настройке инсталляции SQL Remote необходимо выполнить следующие задачи:

- ◆ **Подготовка сервера для SQL Remote.** Для того чтобы Adaptive Server Enterprise работал как узел SQL Remote, необходимо его соответствующим образом сконфигурировать. В частности, сюда входит инсталляция системных объектов SQL Remote и системных объектов очереди с сохранением.
- ◆ **Выбор типов сообщений.** Необходимо выбрать применяемый способ обмена информацией: совместный доступ к файлам, электронная почта, другие типы сообщений или комбинированный способ.
- ◆ **Обеспечение надлежащих полномочий.** Каждому пользователю системы SQL Remote должны быть предоставлены полномочия на доступ как к своей собственной, так и к консолидированной базе данных.
- ◆ **Извлечение удаленных баз данных.** Из консолидированной базы данных необходимо извлечь начальную копию удаленной базы данных для каждой удаленной базы данных.

В данной главе описаны все эти задачи.

Все действия по администрированию выполняются в консолидированной базе данных

Настройка SQL Remote, как и все задачи по администрированию, выполняются в консолидированной базе данных администратором базы данных или системным администратором.

Все задачи, связанные с конфигурированием SQL Remote, выполняются системным администратором Sybase. Дополнительную информацию о среде Adaptive Server Enterprise см. в документации по Adaptive Server Enterprise.

Подготовка сервера Adaptive Server Enterprise

Перед началом работы

В данном разделе предполагается, что следующие задачи уже выполнены:

- ◆ Сервер Adaptive Server Enterprise, который должен содержать базу данных SQL Remote, уже установлен.
- ◆ На компьютере установлено программное обеспечение SQL Remote. Для установки программного обеспечения SQL Remote запустите программу установки с CD-ROM.
- ◆ На сервере Adaptive Server Enterprise создана база данных, которая будет задействована в репликации SQL Remote.
- ◆ Имеются полномочия системного администратора на управление работой сервера Adaptive Server Enterprise, а также полномочия владельца базы данных.

Обеспечение достаточного размера временной базы данных TEMPDB

SQL Remote использует временную базу данных TEMPDB для следующих целей:

- ◆ В утилите извлечения базы данных, служащей для создания удаленных баз данных, база данных TEMPDB используется для хранения временного набора системных таблиц Adaptive Server Anywhere.
- ◆ При подключении к серверу Message Agent создает временную таблицу с именем **#remote**.

Поэтому размер база данных TEMPDB должен превышать 2 Мб (размер по умолчанию). Требуемый размер базы данных зависит от количества таблиц и столбцов в системе SQL Remote; как правило, достаточно 10 Мб.

Установка системных объектов SQL Remote

В базе данных на сервере Adaptive Server Enterprise, которая будет задействована в репликации SQL Remote, необходимо установить определенное количество системных таблиц, представлений и хранимых процедур SQL Remote.

❖ Процедура установки системных объектов SQL Remote

- 1 В папке установки SQL Remote найдите сценарий инициализации SQL Remote - *ssremote.sql*.
- 2 Создайте резервную копию файла сценария *ssremote.sql*. Затем добавьте следующие две строки в начало *ssremote.sql*:

```
use имя_БД
go
```

где *имя_БД* - имя базы данных, которая будет входить в устанавливаемую систему репликации SQL Remote.

С помощью этих двух строк база данных *имя_БД* становится текущей базой данных, в которой и будут создаваться таблицы SQL Remote. Владелец таблиц SQL Remote является владелец этой базы данных.

- 3 Запустите сценарий на сервере Adaptive Server Enterprise.

Войдите в папку, содержащую файл сценария, и для запуска сценария введите следующую команду (вся команда вводится в одной строке):

```
isql -S имя-сервера -U идентификатор_пользователя -P  
пароль -i ssremote.sql -o файл-журнала
```

где *имя-сервера* - имя сервера Adaptive Server Enterprise; *идентификатор_пользователя* и *пароль* – имя и пароль пользователя с полномочиями системного администратора, который является владельцем базы данных на сервере; *файл-журнала* - имя файла журнала для хранения информации о выполнении сценария.

☞ *идентификатор_пользователя* должен соответствовать имени, используемому в Message Agent. Для получения дополнительной информации см. раздел "Message Agent и безопасность репликации" на стр. 211.

- 4 Просмотрите файл журнала и убедитесь в отсутствии ошибок в созданных таблицах и процедурах.

При выполнении сценария в базе данных создается набор системных объектов SQL Remote.

Системные объекты SQL Remote

При выполнении сценария в базе данных создаются следующие объекты:

- ◆ **Системные таблицы SQL Remote.** Набор таблиц, в которых хранится информация SQL Remote. Имена этих таблиц начинаются с **sr_**.
- ◆ **Системные представления SQL Remote.** Набор представлений, содержащих информацию SQL Remote в более удобной для прочтения форме. Имена этих представлений начинаются с **sr_**, а заканчиваются на **s**.
- ◆ **Системные процедуры SQL Remote.** Набор хранимых процедур, используемых для выполнения задач по администрированию и конфигурированию SQL Remote. Имена этих процедур начинаются с **sp_**, что указывает на их роль в управлении системой.

Предостережение: не редактируйте системные таблицы SQL Remote

Ни при каких обстоятельствах не вносите изменения в системные таблицы SQL Remote напрямую. Эти действия могут привести к появлению недопустимых данных в таблице, что приведет к невозможности нормального функционирования SQL Remote. Все задачи по администрированию системы выполняются с помощью системных процедур SQL Remote.

Установка очереди с сохранением из командной строки

Очередь с сохранением – это пара таблиц базы данных для хранения транзакций, используемых в системе репликации. Когда информация о транзакциях больше не нужна, она удаляется из этих таблиц. Очередь с сохранением необходима каждой базе данных Adaptive Server Enterprise, которая будет входить в систему SQL Remote.

☞ Для получения дополнительной информации об очереди с сохранением см. раздел "Очередь с сохранением" на стр. 231.

Очередь с сохранением может располагаться в той же базе данных, что и база данных, которая будет задействована в репликации SQL Remote, либо в отдельной базе данных. Если очередь с сохранением содержится в отдельной базе данных, это может усложнить схему резервного копирования и восстановления данных. С другой стороны, в этом случае появляется возможность повышения

производительности путем переноса нагрузки очереди с сохранением на отдельные устройства и/или на отдельный сервер Adaptive Server Enterprise.

❖ Процедура установки очереди с сохранением

- 1 В папке установки SQL Remote найдите сценарий инициализации очереди с сохранением - *stable.sql*.

- 2 Создайте резервную копию файла сценария *stableq.sql*. Затем добавьте следующие две строки в начало *stableq.sql*:

```
use имя_БД
go
```

где *имя_БД* - имя базы данных, в которой будет содержаться очередь с сохранением.

С помощью этих двух строк база данных *имя_БД* становится текущей базой данных, в которой и будет создаваться очередь с сохранением. Владельцем таблиц очереди с сохранением является владелец этой базы данных.

- 3 Запустите сценарий на сервере Adaptive Server Enterprise.

Войдите в папку, содержащую сценарий очереди сохранения, и для запуска сценария введите следующую команду (вся команда вводится в одной строке):

```
isql -S имя-сервера -U идентификатор_пользователя -P
пароль -i STABLEQ.SQL -o файл-журнала
```

где *имя-сервера* - имя сервера Adaptive Server Enterprise; *идентификатор_пользователя* и *пароль* – имя и пароль пользователя с полномочиями системного администратора, который является владельцем базы данных на сервере; *файл-журнала* - имя файла журнала для хранения информации о выполнении сценария.

☞ *идентификатор_пользователя* должен соответствовать имени, используемому в Message Agent. Для получения дополнительной информации см. раздел "Message Agent и безопасность репликации" на стр. 211.

- 4 Просмотрите файл журнала и убедитесь в отсутствии ошибок в созданных таблицах и процедурах.

Обновление SQL Remote для Adaptive Server Enterprise

В данном разделе описывается процедура обновления SQL Remote для Adaptive Server Enterprise.

Из-за того, что в систему SQL Remote может входить большое количество баз данных, не рекомендуется обновлять программное обеспечение одновременно на всех компьютерах. В системе SQL Remote предусмотрена возможность поэтапного обновления. При обновлении программного обеспечения SQL Remote порядок, в котором обслуживаются компьютеры, не имеет значения, поскольку формат передаваемых сообщений совместим с предыдущими версиями.

❖ Процедура обновления SQL Remote

- 1 Создайте резервные копии консолидированной базы данных и базы данных очереди с сохранением (если очередь с сохранением располагается в отдельной БД).
- 2 Установите новое программное обеспечение SQL Remote для Adaptive Server Enterprise.
- 3 Запустите сценарий *ssupdate.sql* в консолидированной базе данных для обновления системных таблиц и процедур SQL Remote.
Сценарий *ssupdate.sql* хранится в папке Sybase.
- 4 Запустите сценарий *squpdate.sql* в базе данных очереди с сохранением для обновления таблиц и процедур очереди с сохранением SQL Remote.

Сценарий *squpdate.sql* хранится в папке Sybase.

Обновление программного обеспечения завершено.

Деинсталляция SQL Remote

В данном разделе описан процесс удаления объектов SQL Remote и очереди с сохранением из соответствующих баз данных.

❖ Процедура деинсталляции объектов SQL Remote из базы данных

- 1 Выполните подключение к базе данных, содержащей объекты SQL Remote, как пользователь с правами dbo.
- 2 Выполните хранимую процедуру **sp_drop_sql_remote** для удаления всех объектов SQL Remote, кроме самой процедуры. Процедура **sp_drop_sql_remote** устанавливается вместе с другими объектами SQL Remote.

```
exec sp_drop_sql_remote  
go
```

- 3 Для завершения деинсталляции удалите процедуру **sp_drop_sql_remote**.

```
drop procedure sp_drop_sql_remote  
go
```

❖ Процедура деинсталляции очереди с сохранением из базы данных

- 1 Выполните подключение к базе данных, содержащей очередь с сохранением, как пользователь с правами dbo.
- 2 Выполните хранимую процедуру **sp_queue_drop** для удаления всех объектов очереди с сохранением, кроме самой процедуры. Процедура **sp_queue_drop** устанавливается вместе с другими объектами очереди с сохранением.

```
exec sp_queue_drop  
go
```

- 3 Для завершения деинсталляции удалите процедуру **sp_queue_drop**.

```
drop procedure sp_queue_drop  
go
```


ГЛАВА 4

Учебные разделы для пользователей Adaptive Server Anywhere

Об этой главе

В данной главе описан процесс настройки простой системы репликации с использованием Adaptive Server Anywhere.

Содержание

Раздел	Страница
Введение	26
Учебный раздел: репликация Adaptive Server Anywhere с использованием Sybase Central	29
Учебный раздел: репликация Adaptive Server Anywhere с использованием Interactive SQL и dbxtract	36
Запуск репликации данных	42
Пример публикации	45

Введение

В нижеприведенных учебных разделах описывается процесс настройки простой системы репликации SQL Remote с помощью Adaptive Server Anywhere.

Цели

В данных учебных разделах пользователь действует как системный администратор консолидированной базы данных Adaptive Server Anywhere и выполняет настройку простой системы репликации. В систему репликации входит простая база данных продаж, содержащая две таблицы.

В консолидированную базу данных входит вся вышеуказанная БД, тогда как удаленная база данных содержит одну таблицу целиком и несколько строк другой таблицы.

В учебных разделах рассматриваются следующие операции:

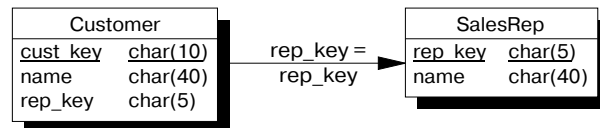
- ◆ Создание консолидированной базы данных на сервере Adaptive Server Anywhere;
- ◆ Создание системы репликации с совместным доступом к файлам с одной удаленной базой данных Adaptive Server Anywhere;
- ◆ Репликация данных между двумя этими базами данных.

База данных

В учебных разделах используется простая база данных с двумя таблицами. В одной таблице содержится информация о торговых представителях, а в другой - о клиентах. Эти таблицы намного проще по своей структуре тех таблиц, которые использовались бы в реальной базе данных, однако такое упрощение позволяет более подробно рассмотреть аспекты, связанные с репликацией.

Схема базы данных

Схема базы данных, представленной в учебных разделах, показана на следующем рисунке.



Обратите внимание на следующее:

- ◆ Каждому торговому представителю соответствует одна строка в таблице **SalesRep**.
- ◆ Аналогично, каждому клиенту соответствует одна строка в таблице **Customer**.
- ◆ Каждый клиент имеет своего торгового представителя. Это назначение реализуется в базе данных посредством внешнего ключа (rep_key) таблицы **Customer** к таблице **SalesRep**. Между таблицами **Customer** и **SalesRep** существует связь типа "один ко многим".

Таблицы в базе данных

Ниже приведено подробное описание используемых таблиц:

Таблица	Описание
SalesRep	<p>Каждая строка соответствует одному торговому представителю, работающему в компании. В таблице SalesRep имеются следующие столбцы:</p> <ul style="list-style-type: none"> ◆ rep_key - идентификатор торгового представителя. Этот столбец является первичным ключом. ◆ name - имя торгового представителя. <p>Эта таблица создается с помощью следующего оператора SQL:</p> <pre>CREATE TABLE SalesRep (rep_key CHAR(12) NOT NULL, name CHAR(40) NOT NULL, PRIMARY KEY (rep_key) }</pre>
Customer	<p>Каждая строка соответствует одному клиенту, ведущему дела с данной компанией. В таблицу Customer входят следующие столбцы:</p> <ul style="list-style-type: none"> ◆ cust_key - идентификатор клиента. Этот столбец является первичным ключом. ◆ name - имя клиента. ◆ rep_key - идентификатор торгового представителя, который работает с данным клиентом. Этот столбец является внешним ключом к таблице SalesRep. <p>Эта таблица создается с помощью следующего оператора SQL:</p> <pre>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))</pre>

Цели репликации

Цель производимой репликации данных заключается в обеспечении каждого торгового представителя следующей информацией:

- ◆ Полная таблица **SalesRep**;
- ◆ Информация о назначенных торговым представителям клиентах.

В учебных разделах описан процесс достижения этих целей при помощи SQL Remote.

Утилиты Sybase Central и утилиты командной строки

Использование Sybase Central или командной строки

Материал в учебных разделах представлен дважды. В одном учебном разделе описывается настройка инсталляции при помощи утилиты управления Sybase Central. Во втором учебном разделе описывается настройка инсталляции при помощи утилит командной строки, где каждую команду необходимо вводить вручную.

Что дальше?

- ◆ Для перехода к учебному разделу по репликации с использованием Sybase Central см. раздел "Учебный раздел: репликация Adaptive Server Anywhere с использованием Sybase Central" на стр. 29.
- ◆ Для перехода к учебному разделу по репликации с непосредственным вводом команд см. раздел "Учебный раздел: репликация Adaptive Server Anywhere с использованием Interactive SQL и dbxtract" на стр. 36.

Учебный раздел: репликация Adaptive Server Anywhere с использованием Sybase Central

Далее представлены учебные разделы, в которых описаны процедуры настройки простой системы репликации SQL Remote в Adaptive Server Anywhere при помощи Sybase Central.

При использовании Sybase Central для выполнения задач по администрированию SQL Remote вводить операторы SQL необязательно. Учебный раздел для пользователей, не имеющих доступа к Sybase Central или предпочитающих использовать утилиты командной строки, приведен на стр. 36, "Учебный раздел: репликация Adaptive Server Anywhere с использованием Interactive SQL и dbxtract", в котором описаны операторы SQL, неявно выполняемые Sybase Central.

В данном учебном разделе пользователь действует как системный администратор консолидированной базы данных Adaptive Server Anywhere и выполняет настройку простой системы репликации через систему передачи сообщений путем совместного доступа к файлам. В рассматриваемом примере используется упрощенная модель системы автоматизации торговой деятельности. Имеются две таблицы, в одной из которых содержится список торговых представителей, а в другой - список клиентов. В процессе настройки производится репликация данных этих таблиц между одной консолидированной базой данных и одной удаленной базой данных. Для выполнения описываемых процедур необходим один компьютер с установленным ПО.

Этот учебный раздел предполагает наличие определенных знаний о принципах работы Sybase Central. Вводная информация о Sybase Central представлена в разделе "Учебный раздел: управление базами данных при помощи Sybase Central" (Tutorial: Managing Databases with Sybase Central) на стр. 49 в документе "Введение в SQL Anywhere Studio" (Introducing SQL Anywhere Studio).

Подготовка к работе с учебным разделом по репликации с использованием Sybase Central

В данном разделе описаны шаги, которые необходимо выполнить для подготовки к работе с обучающими материалами. Сюда входит следующее:

- ◆ Создание папок и баз данных, необходимых для выполнения описанных в учебном разделе процедур;
- ◆ Добавление таблиц в консолидированную базу данных.

❖ Процедуры подготовки к работе с учебным разделом

- 1 Создайте папку для хранения файлов, создаваемых во время обучения, например, *c:\tutorial*.

```
mkdir c:\tutorial
```

- 2 Создайте в системе репликации подпапку для каждого из двух идентификаторов пользователей для хранения сообщений этих пользователей. Эти подпапки можно создать путем ввода нижеприведенных операторов в системной командной строке:

```
mkdir c:\tutorial\HQ
```

```
mkdir c:\tutorial\field
```

- 3 Создайте базу данных **HQ**:

- ◆ Запустите Sybase Central.

- ◆ В левой области окна откройте папку Utilities Adaptive Server Anywhere.
- ◆ Дважды щелкните по пункту Create Database в правой области окна. Появится мастер создания базы данных (Create Database).
- ◆ Создайте базу данных с именем файла `c:\tutorial\HQ.db`.
- ◆ Для этой базы данных используйте установки по умолчанию.

База данных Adaptive Server Anywhere – это обычный файл, который, при необходимости, можно скопировать в другие местоположения и на другие компьютеры.

Далее необходимо добавить пару таблиц в консолидированную базу данных.

❖ Процедура добавления таблиц в консолидированную базу данных

- 1 Выполните подключение к базе данных **HQ** из Sybase Central с использованием идентификатора пользователя **DBA** и пароля **SQL**.
- 2 Щелкните по папке Tables базы данных **HQ**.
- 3 Дважды щелкните по пункту Add Table и создайте таблицу с именем **SalesRep** и следующими столбцами:

Ключ	Столбец	Тип данных	Размер/Условие
Первичный ключ	Rep_key	Char	5
	Name	Char	40

Вводить данные в окне свойств столбца (Column property) или диалоге расширенных свойств таблицы (Advanced Table Properties) не требуется.

- 4 Сохраните таблицу и закройте диалог.
- 5 Дважды щелкните по пункту Add Table и создайте таблицу с именем **Customer** и следующими столбцами:

Ключ	Столбец	Тип данных	Размер/Условие
Первичный ключ	Cust_key	Char	10
	Name	Char	40
	Rep_key	Char	5

Вводить данные в окнах свойств не требуется.

- 6 Сохраните таблицу и закройте диалог.
- 7 В папке Tables откройте контейнер таблицы Customer, после чего откройте папку Foreign Keys в этой таблице.
- 8 Дважды щелкните по пункту Add Foreign Key. С помощью этого мастера добавьте внешний ключ к столбцу **Rep_key** таблицы **SalesRep**. Для данного внешнего ключа можно использовать установки по умолчанию.

Подготовка к работе с учебным разделом завершена.

Настройка консолидированной базы данных

В данном подразделе учебного раздела описывается процесс подготовки консолидированной базы данных простой системы репликации.

Процедура подготовки консолидированной базы данных для репликации включает следующие шаги:

- 1 Создание типа сообщений для использования при репликации.
- 2 Предоставление полномочий PUBLISH соответствующему пользователю для определения источника исходящих сообщений.
- 3 Предоставление полномочий REMOTE для всех пользователей-получателей сообщений.
- 4 Создание публикации с описанием реплицируемых данных.
- 5 Создание подписки с описанием того, кто будет получать публикации.

Для выполнения этих задач требуется наличие полномочий администратора БД.

Добавление типа сообщений SQL Remote

Для всех сообщений, передаваемых в ходе репликации, должен быть установлен тип сообщения. Описание типа сообщений состоит из двух частей:

- ◆ Система передачи сообщений, поддерживаемый SQL Remote. В данном учебном разделе используется система FILE.
- ◆ Адрес для этой системы передачи сообщений с целью определения источника исходящих сообщений.

В базах данных Adaptive Server Anywhere типы сообщений уже созданы, однако необходимо указать адрес для используемого при репликации типа сообщений.

❖ Процедура добавления адреса к типу сообщений

- 1 Выполните подключение к базе данных HQ из Sybase Central.
- 2 Откройте папку SQL Remote для базы данных HQ.
- 3 В папке SQL Remote откройте папку Message Types.
- 4 В правой области окна щелкните правой кнопкой мыши по типу сообщений FILE и во всплывающем меню выберите пункт Properties.
- 5 Введите адрес издателя публикации, который будет являться адресом возврата для удаленных пользователей. Введите имя созданной папки для хранения сообщений, предназначенных для консолидированной базы данных (hq).

Адрес указывается относительно переменной среды SQLRemote или записи в реестре. Поскольку это значение не было задано, укажите адрес относительно папки, из которой запускается Message Agent. Для обеспечения правильной интерпретации адресов Message Agent необходимо запускать из папки учебного раздела.

☞ Для получения информации об установке значения переменной SQLRemote см. раздел "Установка параметров управления типами сообщений" на стр. 186.

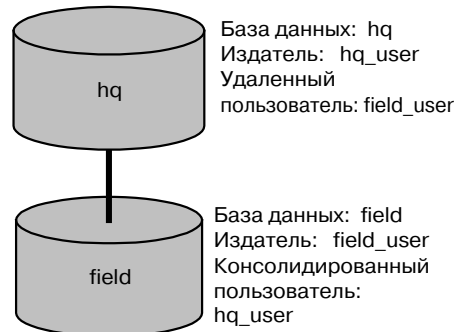
- 6 Нажмите ОК для сохранения типа сообщений.

Добавление к базе данных издателя и удаленного пользователя

В иерархической системе репликации SQL Remote для каждой базы данных может существовать максимум одна БД верхнего уровня (консолидированная база данных) и несколько БД нижнего уровня (удаленные базы данных). Также возможна ситуация отсутствия БД верхнего или нижнего уровня.

В данном учебном разделе рассматривается система из двух уровней: текущей базой данных является консолидированная база данных, для которой БД верхнего уровня нет, а на нижнем уровне имеется только одна удаленная база данных.

Имеющиеся две базы данных представлены на следующей диаграмме:



Для каждой базы данных при настройке системы репликации SQL Remote доступны три вида полномочия, предоставляемых в целях идентификации баз данных в иерархической системе:

- ◆ **Полномочия PUBLISH** - идентифицирует текущую базу данных во всех исходящих сообщениях;
- ◆ **Полномочия REMOTE** – идентифицирует каждую базу данных на нижнем уровне иерархии, получающую сообщения из текущей базы данных;
- ◆ **Полномочия CONSOLIDATE** - идентифицирует базу данных на верхнем уровне, получающую сообщения из текущей базы данных.

Эти полномочия могут предоставляться только пользователем, обладающим полномочиями администратора БД. Для выполнения следующих примеров необходимо выполнить подключение к базе данных **hq** из Sybase Central с использованием идентификатора пользователя **DBA** и пароля **SQL**.

Добавление
издателя для базы
данных

Любая консолидированная или удаленная база данных, из которой осуществляется тиражирование изменений данных в другие базы данных системы репликации, называется базой данных издателя. Идентификация каждой базы данных в системе репликации производится с помощью одного идентификатора пользователя. Идентификатор для БД издателя задается путем добавления издателя для этой базы данных. В данном разделе описана процедура установки полномочий для консолидированной базы данных **hq**.

Сначала создайте идентификатор пользователя **hq_user**, который будет являться идентификатором издателя.

❖ **Процедура создания нового пользователя-издателя**

- 1 Откройте папку Users & Groups.
- 2 Дважды щелкните по пункту Add User. Появится мастер создания пользователя (User Creation).
- 3 Введите имя **hq_user** и пароль **hq_pwd** и нажмите кнопку Finish.
- 4 Щелкните правой кнопкой мыши по значку **hq_user** и выберите во всплывающем меню пункт Change to Publisher.

База данных может иметь только одного издателя. Выяснить, какой пользователь является издателем, можно путем открытия папки SQL Remote.

Добавление
удаленного
пользователя

Идентификация каждой удаленной базы данных в консолидированной БД производится по идентификатору пользователя с полномочиями REMOTE. Независимо от того, находится ли удаленная база данных на персональном сервере БД или на сетевом сервере с большим количеством пользователей, для ее

идентификации в консолидированной базе данных необходим один идентификатор пользователя.

При настройке мобильной рабочей группы удаленные пользователи могут уже являться пользователями консолидированной базы данных, и в этом случае добавлять новых пользователей не требуется; однако возможно, что будет необходимо задать их статус как удаленных пользователей.

При добавлении к базе данных удаленного пользователя вместе с идентификатором этого пользователя необходимо указать используемую им систему передачи сообщений и адрес этой системы.

❖ Процедура добавления удаленного пользователя

- 1 Откройте папку Remote Users (эта папка находится в папке SQL Remote).
- 2 Дважды щелкните по пункту Add Remote User.
- 3 Создайте удаленного пользователя с идентификатором пользователя **field_user**. Нажмите кнопку Next.
- 4 Введите пароль **field_pwd** и нажмите Next.
- 5 Выберите тип сообщений **file** и введите адрес **field**.

Так же, как и адрес издателя, адрес удаленного пользователя указывается относительно переменной среды SQLREMOTE или записи в реестре. Поскольку это значение не было задано, укажите адрес относительно папки, из которой запускается Message Agent. Для обеспечения правильной интерпретации адресов Message Agent необходимо запускать из папки учебного раздела.

☞ Для получения информации об установке значения переменной SQLRemote см. раздел "Установка параметров управления типами сообщений" на стр. 186.

- 6 В следующем окне отметьте опцию Send Then Close. (Во многих средах выполнения опция Send Then Close не устанавливается, но для данного учебного раздела рекомендуется ее установить.)
- 7 В следующем окне выберите полномочия Remote DBA с тем, чтобы пользователь мог запустить Message Agent.
- 8 По завершении установки параметров нажмите кнопку Finish для создания удаленного пользователя.

Создание пользователей, которые будут работать с системой, завершено.

Добавление публикаций и подписок

В данном разделе описан процесс добавления публикации в базу данных, а также процесс добавления пользовательской подписки на эту публикацию. С помощью данной публикации производится репликация всех строк таблицы **SalesRep** и некоторых строк таблицы **Customer**.

❖ Процедура добавления публикации

- 1 Щелкните по папке Publications в папке SQL Remote.
- 2 Дважды щелкните по пункту Add Publication. Появится мастер создания публикации (Publication Creation).
- 3 В первом окне мастера введите имя публикации - **SalesRepData**.
- 4 На закладке Tables в следующем окне выберите **SalesRep** из списка Matching Tables. Нажмите кнопку Add.

В списке Selected Tables справа появится нужная таблица.

Добавление подписки

- 5 Из списка Matching Tables выберите пункт **Customer**. Нажмите кнопку Add.
- 6 Перейдите на закладку Subscribe By. На этой закладке выберите таблицу Customer и введите выражение **rep_key**. Для завершения процесса создания публикации нажмите кнопку Finish.

Каждый пользователь, который будет получать информацию об изменениях в публикации, должен быть **подписан** на эту публикацию. Подписки могут создаваться только для действительных удаленных пользователей. В рассматриваемом примере требуется добавить подписку на публикацию **SalesRepData** для пользователя удаленной базы данных **field_user**.

❖ Процедура добавления подписки

- 1 Откройте папку Publications (эта папка находится в папке SQL Remote).
- 2 Щелкните правой кнопкой по публикации **SalesRepData** и выберите во всплывающем меню пункт Properties.
- 3 Перейдите на закладку SQL Remote Subscriptions.
- 4 Нажмите кнопку Subscribe и выберите подписку для пользователя **field_user**. В качестве значения подписки (Subscription value) введите **rep1** и нажмите ОК.

Значение подписки – это выражение, которое соответствует выражению Subscribe By в публикации. При выполнении следующих шагов идентификатору пользователя **field_user** назначается ключ **rep_key** от **rep1**.

Настройка консолидированной базы данных завершена.

Настройка удаленной базы данных в Sybase Central

Необходимо создать и сконфигурировать удаленную базу данных, которая будет участвовать в обмене сообщениями и использоваться при настройке SQL Remote.

Как и для консолидированной базы данных, для удаленной БД необходимо назначить издателя (в данном случае с идентификатором пользователя **field_user**) с целью идентификации источника исходящих сообщений. Также необходимо определить пользователя **hq_user** как пользователя с консолидированными полномочиями. Потребуется создать публикацию **SalesRepData** и подписку для идентификатора пользователя **hq_user**.

Удаленную базу данных необходимо **синхронизировать** с консолидированной базой данных, т.е. для запуска процесса репликации удаленная база данных должна получить копию актуальных данных. В данном случае в публикацию данные еще не занесены.

Утилита извлечения базы данных позволяет выполнить все шаги по созданию удаленной базы данных вместе с подписками и необходимыми идентификаторами пользователей.

Для удаленного пользователя **field_user** необходимо извлечь эту базу данных из консолидированной базы данных.

❖ Процедура извлечения базы данных

- 1 Выполните подключение к базе данных **HQ**.
- 2 Щелкните правой кнопкой по базе данных и выберите во всплывающем меню пункт Extract Database.
- 3 В первом окне мастера нажмите кнопку Next.
- 4 Выберите извлечение базы данных HQ.

- 5 Выберите извлечение уровня изоляции 3.
- 6 Выберите пункт Start Subscriptions Automatically для пользователя **field_user**.
- 7 При выборе местоположения файла перезагрузки оставьте значения по умолчанию и выберите извлечение структуры и данных.
- 8 Выберите извлечение всех частей схемы.
- 9 При выборе местоположения для сохранения оставьте значения по умолчанию.
- 10 Откажитесь от извлечения полных определений публикации.
- 11 Создайте базу данных как файл `c:\tutorial\field.db` и нажмите кнопку Finish для завершения создания удаленной базы данных.

При правильной настройке SQL Remote удаленную базу данных **field** потребуется загрузить в компьютер вместе с сервером Adaptive Server Anywhere и всеми необходимыми клиентскими приложениями. В этом учебном разделе загрузка базы данных не выполняется, а для ввода и репликации данных используется Interactive SQL.

Необходимо выполнить подключение к базе данных **field** в качестве администратора БД и удостовериться, что созданы все необходимые объекты базы данных. Сюда входят таблицы **SalesRep** и **Customer**, публикация **SalesRepData** и подписка для консолидированной базы данных.

Что дальше?

Теперь система готова к репликации.

☞ Для перехода к следующему шагу – вставке и репликации данных – см. раздел "Запуск репликации данных" на стр. 42.

Учебный раздел: репликация Adaptive Server Anywhere с использованием Interactive SQL и dbxtract

Далее приведен учебный раздел, в котором описывается настройка простой системы репликации SQL Remote для пользователей, предпочитающих работать с командной строкой, либо для тех, кто хочет ознакомиться с процессами, неявно выполняемыми в Sybase Central.

В данном учебном разделе описаны операторы SQL для управления SQL Remote, которые можно запускать из Interactive SQL. Также здесь описан процесс запуска утилиты командной строки *dbxtract* для извлечения удаленных баз данных из консолидированной базы данных.

В данном учебном разделе пользователь действует как системный администратор консолидированной базы данных и выполняет настройку простой системы репликации через систему передачи сообщений путем совместного доступа к файлам. В рассматриваемом примере используется упрощенная модель системы автоматизации торговой деятельности. Имеются две таблицы, в одной из которых содержится список торговых представителей, а в другой - список клиентов. В процессе настройки производится репликация данных этих таблиц между одной консолидированной базой данных и одной удаленной базой данных. Для выполнения описываемых процедур необходим один компьютер с установленным ПО.

Подготовка к работе с учебным разделом по репликации

В данном разделе описаны шаги, которые необходимо выполнить для подготовки к работе с обучающими материалами. Сюда входит следующее:

- ◆ Создание папок и баз данных, необходимых для выполнения описанных в учебном разделе процедур;
- ◆ Добавление таблиц в консолидированную базу данных.

❖ Процедура создания баз данных и папок для работы с учебным разделом

- 1 Создайте папку для хранения файлов, создаваемых во время обучения, например, *c:\tutorial*.

```
mkdir c:\tutorial
```

- 2 В учебном разделе используются две базы данных: консолидированная база данных с именем *hq.db* и удаленная база данных с именем *field.db*. Откройте папку учебного раздела и создайте эти базы данных путем ввода в командной строке следующих операторов:

```
dbinit hq.db
```

```
dbinit field.db
```

- 3 Создайте подпапки для каждого из двух идентификаторов пользователя в системе репликации. Создание подпапок осуществляется путем ввода в командной строке следующих операторов:

```
mkdir c:\tutorial\hq
```

```
mkdir c:\tutorial\field
```

Следующим шагом является добавление пары таблиц в консолидированную базу данных.

❖ Процедура добавления таблиц в консолидированную базу данных

1 Выполните подключение к базе данных *hq.db* из Interactive SQL с использованием идентификатора пользователя **DBA** и пароля **SQL**.

2 Для создания таблицы **SalesRep** выполните следующий оператор CREATE TABLE:

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

3 Для создания таблицы **Customer** выполните следующий оператор CREATE TABLE:

```
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 )  
);
```

Подготовка к работе с учебным разделом завершена.

Настройка консолидированной базы данных

В данном подразделе учебного раздела описывается процесс подготовки консолидированной базы данных простой системы репликации.

Для выполнения этой задачи требуется наличие полномочий администратора БД.

Создание типа сообщений SQL Remote

Для всех сообщений, передаваемые в ходе репликации, должен быть установлен тип сообщения. Описание типа сообщений состоит из двух частей:

- ◆ Система передачи сообщений, поддерживаемый SQL Remote. В данном учебном разделе используется система FILE;
- ◆ Адрес для этой системы передачи сообщений с целью определения источника исходящих сообщений.

❖ Процедура создания типа сообщений

◆ В Interactive SQL создайте тип сообщений file с помощью следующего оператора:

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

Адресом (**hq**) для системы передачи file является папка, в которой помещаются файлы, содержащие сообщения. Адрес указывается относительно переменной среды SQLRemote или записи в реестре. Поскольку это значение не было задано, укажите адрес относительно папки, из которой запускается Message Agent. Для обеспечения правильной интерпретации адресов Message Agent необходимо запускать из папки учебного раздела.

☞ Для получения информации об установке значения переменной SQLRemote см. раздел "Установка параметров управления типами сообщений" на стр. 186.

Предоставление полномочий PUBLISH и REMOTE в консолидированной базе данных

В иерархической системе репликации SQL Remote для каждой базы данных может существовать максимум одна БД верхнего уровня иерархии и несколько БД нижнего уровня (удаленные базы данных).

Полномочия PUBLISH используются для идентификации текущей базы данных во всех исходящих сообщениях, а полномочия REMOTE - для идентификации каждой базы данных, получающей сообщения из текущей базы данных.

Эти полномочия могут предоставляться только пользователем, обладающим полномочиями администратора БД. Для выполнения следующих примеров необходимо выполнить подключение к базе данных **hq** с помощью утилиты Interactive SQL с использованием идентификатора пользователя **DBA** и пароля **SQL**.

Предоставление полномочий PUBLISH для идентификации источника исходящих сообщений (GRANT PUBLISH)

Любая база данных, из которой осуществляется тиражирование изменений данных в другие базы данных системы репликации, называется базой данных издателя. Идентификация баз данных, которые осуществляют публикацию изменений, производится в системе репликации с помощью соответствующих идентификаторов пользователей. Такой идентификатор для базы данных издателя задается при помощи оператора GRANT PUBLISH. В данном разделе описана процедура установки полномочий для консолидированной базы данных (*hq.db*).

❖ Процедура создания издателя для базы данных

- ◆ Выполните подключение к базе данных из Interactive SQL и введите следующий оператор:

```
GRANT CONNECT
TO hq_user
IDENTIFIED BY hq_pwd ;

GRANT PUBLISH TO hq_user ;
```

Идентификатор пользователя-издателя базы данных можно в любой момент просмотреть с помощью специального постоянного выражения CURRENT PUBLISHER:

```
SELECT CURRENT PUBLISHER
```

Предоставление полномочий REMOTE для идентификации базы данных-получателя сообщений (GRANT REMOTE)

Идентификация каждой удаленной базы данных в консолидированной БД производится с помощью оператора GRANT REMOTE. Независимо от того, находится ли удаленная база данных на персональном сервере БД или на сетевом сервере с большим количеством пользователей, для ее идентификации в консолидированной базе данных необходим один идентификатор пользователя.

При настройке мобильной рабочей группы удаленные пользователи могут уже являться пользователями консолидированной базы данных, и в этом случае со стороны администратора БД никаких дополнительных действий не требуется.

Оператор GRANT REMOTE используется для определения системы передачи сообщений, которая используется при отправке сообщений адресату, а также ее адреса.

❖ Процедура добавления удаленного пользователя

- ◆ Выполните подключение к базе данных из Interactive SQL и введите следующие операторы:

```
GRANT CONNECT TO field_user
IDENTIFIED BY field_pwd ;

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

В качестве адреса (ADDRESS) в одиночных кавычках указывается папка, используемая для хранения сообщений, предназначенных пользователю

field_user. Адрес указывается относительно переменной среды SQLRemote или записи в реестре. Поскольку это значение не было задано, укажите адрес относительно папки, из которой запускается Message Agent. Для обеспечения правильной интерпретации адресов Message Agent необходимо запускать из папки учебного раздела.

☞ Для получения информации об установке значения переменной SQLRemote см. раздел "Установка параметров управления типами сообщений" на стр. 186.

Создание публикаций и подписок

Публикация создается при помощи оператора `CREATE PUBLICATION`. Это оператор языка определения данных, для выполнения которого требуются полномочия администратора БД. В данном учебном разделе для создания публикации необходимо выполнить подключение к базе данных **hq** с использованием идентификатора пользователя **DBA** и пароля **SQL**.

Создание публикации в консолидированной базе данных

Создайте публикацию с именем **SalesRepData**, с помощью которой производится репликация всех строк таблицы **SalesRep** и некоторых строк таблицы **Customer**.

❖ Процедура создания публикации

- ◆ Выполните подключение к базе данных из Interactive SQL и введите следующий оператор:

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

Создание подписки

Каждый пользователь, который будет получать информацию об изменениях в публикации, должен быть подписан на эту публикацию. Подписки могут создаваться только для пользователей, имеющих полномочия `REMOTE`. Оператор для предоставления таких полномочий `GRANT REMOTE` содержит адрес, который используется при передаче сообщений.

❖ Процедура создания подписки

- ◆ Выполните подключение к базе данных из Interactive SQL и введите следующий оператор:

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

Значение **rep1** – это значение ключа **rep_key**, которое будет присвоено пользователю **field_user** в таблице **SalesRep**.

Полный оператор `CREATE SUBSCRIPTION` позволяет осуществлять управление данными в подписках; пользователи могут получать только некоторые строки из публикации. Для получения дополнительной информации см. раздел "Оператор `CREATE SUBSCRIPTION`" на стр. 307.

Оператор `CREATE SUBSCRIPTION` позволяет идентифицировать подписчиков и определять получаемую ими информацию. Однако данный оператор не производит синхронизацию данных и запуск процесса передачи сообщений.

Настройка удаленной базы данных

Необходимо создать и сконфигурировать удаленную базу данных, которая будет участвовать в обмене сообщениями и использоваться при настройке SQL Remote. Как и для консолидированной базы данных, для удаленной БД необходимо

назначить издателя (CURRENT PUBLISHER) с целью идентификации источника исходящих сообщений. Также необходимо определить эту БД как подписчика в консолидированной базе данных. Для удаленной базы данных также необходимо создать публикацию и подписку для консолидированной базы данных. Удаленную базу данных необходимо **синхронизировать** с консолидированной базой данных, т.е. удаленная база данных должна получить копию актуальных данных для запуска процесса репликации.

Утилита *dbxtract* позволяет выполнить все шаги по созданию удаленной базы данных вместе с подписками и необходимыми идентификаторами пользователей.

Извлечение информации удаленной базы данных

Откройте папку учебного раздела, не закрывая базу данных **hq**.

Для извлечения базы данных пользователя **field_user** из консолидированной базы данных в системной командной строке введите следующую команду (вся команда вводится в одной строке):

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

Параметр `-v` применяется для вывода подробной информации. Это может пригодиться во время разработки.

При использовании этой команды предполагается, что база данных **hq** запущена на сервере по умолчанию. Если база данных не запущена, введите параметр файла базы данных в строке подключения:

```
dbf=hq.db
```

Этот параметр вводится вместо параметра имени базы данных **dbn**.

☞ Для получения подробной информации об утилите *dbxtract* и ее параметрах см. раздел "Утилита извлечения" на стр. 265.

Команда *dbxtract* создает командный файл SQL с именем *reload.sql* в текущей папке и файл данных в папке *c:\tutorial*. При выполнении этой команды также создаются подписки для удаленных пользователей.

Следующим шагом является загрузка этих файлов в удаленную базу данных.

Загрузка информации удаленной базы данных

❖ Процедура загрузки информации базы данных

- 1 Из папки учебного раздела выполните подключение к удаленной базе данных *field.db* из Interactive SQL с использованием идентификатора пользователя **DBA** и пароля **SQL**.
- 2 Запустите командный файл *reload.sql*:

```
READ C:\tutorial\reload.sql
```

Командный файл *reload.sql* выполняет следующие задачи:

- ◆ Создание типа сообщений в удаленной базе данных.
- ◆ Предоставление полномочий PUBLISH и REMOTE для удаленной и консолидированной базам данных соответственно.
- ◆ Создание таблицы в базе данных. Если перед извлечением в таблице содержались какие-либо данные, командный файл перезапишет копию данных в реплицированную таблицу.
- ◆ Создание публикации для идентификации реплицируемых данных.

- ◆ Создание подписки для консолидированной базы данных и активизация подписки.

При подключении к базе данных **field** в качестве администратора БД проверьте наличие созданных таблиц с помощью следующих операторов:

```
SELECT * FROM SalesRep ;
```

```
SELECT * FROM Customer ;
```

Что дальше?

Теперь система готова к репликации.

☞ Для перехода к следующему шагу – вставке и репликации данных – см. раздел "Запуск репликации данных" на стр. 42.

Запуск репликации данных

Теперь у пользователя имеется сконфигурированная система репликации. В данном разделе описывается процесс репликации данных из консолидированной базы данных в удаленную базу данных и из удаленной БД в консолидированную.

Ввод данных в консолидированную базу данных

Для начала необходимо ввести некоторые данные в консолидированную базу данных.

❖ Процедура ввода данных в консолидированную базу данных

1 Выполните подключение к консолидированной базе данных **hq** с помощью утилиты Interactive SQL с использованием идентификатора пользователя **DBA** и пароля **SQL**.

2 Вставьте две строки в таблицу **SalesRep** и подтвердите вставку с помощью следующего оператора:

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

3 Вставьте две строки в таблицу **Customer** и подтвердите вставку с помощью следующего оператора:

```
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 Проверьте корректность ввода данных с помощью следующих операторов:

```
SELECT *
FROM SalesRep;

SELECT *
FROM Customer;
```

Следующим шагом является передача соответствующих строк в удаленную базу данных.

Отправка данных из консолидированной базы данных

Для отправки строк в удаленную базу данных необходимо запустить Message Agent в консолидированной базе данных. Программа **dbremote** выполняет функцию Message Agent для Adaptive Server Anywhere.

❖ Процедура отправки данных в удаленную базу данных

1 В командной строке перейдите в папку учебного раздела. Например:

```
> c:
> cd c:\tutorial
```

2 Для запуска Message Agent в консолидированной базе данных введите в командной строке следующий оператор:

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

При использовании этой команды предполагается, что база данных **hq** запущена на сервере по умолчанию. Если база данных не запущена, введите имя файла базы данных в параметре **dbf** (вместо параметра **dbn**).

☞ Для получения дополнительной информации о параметрах *dbremote* см. раздел "Message Agent" на стр. 9.

- 3 После отправки сообщений нажмите кнопку Shutdown в окне Message Agent для завершения работы программы. По окончании выполнения в окне Message Agent появится сообщение "Execution completed".

Получение данных в удаленной базе данных

Для получения в удаленной БД оператора вставки данных необходимо запустить Message Agent (*dbremote*) в удаленной базе данных.

❖ Процедура получения данных в удаленной базе данных

- 1 В командной строке перейдите в папку учебного раздела. Например:

```
> c:  
> cd c:\tutorial
```

- 2 Для запуска Message Agent в базе данных **field** введите в командной строке следующий оператор:

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

При использовании этой команды предполагается, что база данных **field** запущена на сервере по умолчанию.

☞ Для получения дополнительной информации о параметрах *dbremote* см. в раздел "Message Agent" на стр. 9.

- 3 После получения сообщений нажмите кнопку Shutdown в окне Message Agent для завершения работы программы. По окончании выполнения в окне Message Agent появится сообщение "Execution completed".

Во время работы базы данных в окне Message Agent отображается информация о состоянии. При работе с реальной БД эту информацию можно поместить в файл журнала с целью ее сохранения. Из записей видно, что сначала Message Agent получает сообщение из базы данных **hq**, а затем отправляет ответное сообщение. Возвращаемое сообщение содержит подтверждение успешного приема обновления в процессе репликации; такие подтверждения являются частью системы отслеживания сообщений SQL Remote, обеспечивающей отправку сообщений даже в случае сбоев системы передачи сообщений.

Проверка получения данных

Теперь необходимо выполнить подключение к удаленной базе данных **field** из Interactive SQL и проверить, имеются ли полученные строки в таблицах **SalesRep** и **Customer**.

❖ Процедура проверки получения данных

- 1 Выполните подключение к базе данных **field** из Interactive SQL.
- 2 Просмотрите содержимое таблицы **SalesRep** путем выполнения следующего оператора:

```
SELECT * FROM SalesRep
```

Будет видно, что таблица **SalesRep** содержит обе строки, введенные в консолидированной базе данных. Это вызвано тем, что в публикацию **SalesRepData** были включены все данные из таблицы **SalesRep**.

3. Просмотрите содержимое таблицы **Customer** путем выполнения следующего оператора:

```
SELECT * FROM Customer
```

Будет видно, что таблица **Customer** содержит только одну строку (Ocean Sports), введенную в консолидированной базе данных. Это происходит потому, что в публикации **SalesRepData** были включены только те клиенты, которые назначены указанному в подписке торговому представителю.

Репликация из удаленной базы данных в консолидированную базу данных

Теперь можно попробовать ввести данные в удаленной базе данных и затем передать эти данные в консолидированную БД. Далее представлены только основные шаги этой процедуры.

❖ Процедура репликации данных из удаленной базы данных в консолидированную базу данных

1. Выполните подключение к базе данных **field** из Interactive SQL.
2. Вставьте строку в удаленной базе данных путем выполнения следующего оператора:

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

3. Подтвердите вставку при помощи следующего оператора:

```
COMMIT;
```

4. При запущенной базе данных **field.db** выполните утилиту **dbremote** из командной строки для отправки сообщения в консолидированную базу данных.

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

5. При запущенной базе данных **hq.db** выполните утилиту **dbremote** из командной строки для получения сообщения в консолидированной базе данных:

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

6. Выполните подключение к консолидированной базе данных. Просмотрите содержимое таблицы **Customer** с помощью следующего оператора:

```
SELECT *
FROM Customer
```

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

В этом простом примере средства предотвращения повторения значений первичного ключа не применяются. Однако в SQL Remote такие средства безопасности предусмотрены. Для получения подробной информации см. разделы, посвященные созданию схем репликации SQL Remote.

Пример публикации

Командный файл *salespub.sql* содержит набор операторов, создающий публикацию в демонстрационной базе данных. Эта публикация более подробно иллюстрирует некоторые аспекты, рассматриваемые в учебных разделах.

❖ Процедура добавления публикации в демонстрационную базу данных

- 1 Выполните подключение к демонстрационной базе данных из Interactive SQL.
- 2 В окне SQL Statements выполните следующий оператор:

```
READ path\scripts\salespub.sql,
```

где *path* – папка SQL Anywhere.

Публикация *salespub.sql* добавляет столбцы в некоторые из таблиц демонстрационной базы данных, создает публикацию и подписки, а также добавляет триггеры, устраняющие возможные конфликты при обновлении данных.

ГЛАВА 5

Учебный раздел для пользователей Adaptive Server Enterprise

Об этой главе

Данная глава является учебным разделом, в котором описан процесс создания и настройки простой системы репликации SQL Remote между базой данных Adaptive Server Enterprise и базой данных Adaptive Server Anywhere.

Содержание

Раздел	Страница
Введение	48
Учебный раздел: репликация Adaptive Server Enterprise	50
Запуск репликации данных	57

Введение

Данная глава представляет собой учебный раздел, в котором рассматривается процесс настройки инсталляции SQL Remote. Данная система производит репликацию данных между базой данных Adaptive Server Enterprise (консолидированная база данных) и базой данных Adaptive Server Anywhere (удаленная база данных).

Цели

В данном учебном разделе пользователь действует как системный администратор консолидированной базы данных Adaptive Server Anywhere и выполняет настройку простой системы репликации. В систему репликации входит простая база данных продаж, содержащая две таблицы.

В консолидированную базу данных входит вся вышеуказанная БД, тогда как удаленная база данных содержит одну таблицу целиком и несколько строк другой таблицы.

В данном учебном разделе рассматриваются следующие операции:

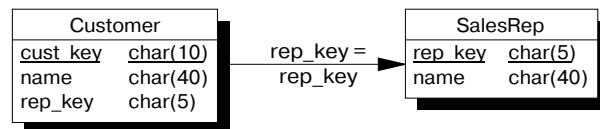
- ◆ Создание консолидированной базы данных на сервере Adaptive Server Enterprise;
- ◆ Создание системы репликации с совместным доступом к файлам с одной удаленной базой данных Adaptive Server Anywhere;
- ◆ Репликация данных между двумя этими базами данных.

База данных

В данном учебном разделе используется простая база данных с двумя таблицами. В одной таблице содержится информация о торговых представителях, а в другой - о клиентах. Эти таблицы намного проще по своей структуре тех таблиц, которые использовались бы в реальной базе данных; однако такое упрощение позволяет более подробно рассмотреть аспекты, связанные с репликацией.

Схема базы данных

Схема базы данных, представленной в учебном разделе, показана на следующем рисунке.



Обратите внимание на следующее:

- ◆ Каждому торговому представителю соответствует одна строка в таблице **SalesRep**.
- ◆ Аналогично, каждому клиенту соответствует одна строка в таблице **Customer**.
- ◆ Каждый клиент имеет своего торгового представителя. Это назначение реализуется в базе данных посредством внешнего ключа (rep_key) таблицы **Customer** к таблице **SalesRep**. Между таблицами **Customer** и **SalesRep** существует связь типа "один ко многим".

Таблицы в базе данных

Ниже приведено подробное описание используемых таблиц:

Таблица	Описание
SalesRep	<p>Каждая строка соответствует одному торговому представителю, работающему в компании. В таблице SalesRep имеются следующие столбцы:</p> <ul style="list-style-type: none"> ◆ rep_key - идентификатор торгового представителя. Этот столбец является первичным ключом. ◆ name - имя торгового представителя. <p>Эта таблица создается с помощью следующего оператора SQL:</p> <pre>CREATE TABLE SalesRep (rep_key CHAR(12) NOT NULL, name CHAR(40) NOT NULL, PRIMARY KEY (rep_key))</pre>
Customer	<p>Каждая строка соответствует одному клиенту, ведущему дела с данной компанией. В таблицу Customer входят следующие столбцы:</p> <ul style="list-style-type: none"> ◆ cust_key - идентификатор клиента. Этот столбец является первичным ключом. ◆ name - имя клиента. ◆ rep_key - идентификатор торгового представителя, который работает с данным клиентом. Этот столбец является внешним ключом к таблице SalesRep. <p>Эта таблица создается с помощью следующего оператора SQL:</p> <pre>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))</pre>

Цели репликации

Цель производимой репликации данных заключается в обеспечении каждого торгового представителя следующей информацией:

- ◆ Полная таблица **SalesRep**;
- ◆ Информация о назначенных торговым представителям клиентах.

В данном учебном разделе описан процесс достижения этих целей при помощи SQL Remote.

Учебный раздел: репликация Adaptive Server Enterprise

Далее представлены учебные разделы, в которых рассматриваются процедуры настройки простой системы репликации SQL Remote.

В данном учебном разделе описаны хранимые процедуры, используемые для конфигурирования и управления SQL Remote. Здесь также описан процесс запуска утилиты *ssxtract* для извлечения удаленных баз данных из консолидированной базы данных и программы Message Agents для передачи информации между базами данных в системе репликации.

В данном учебном разделе пользователь действует как системный администратор консолидированной базы данных и выполняет настройку простой системы репликации через систему передачи сообщений путем совместного доступа к файлам. В рассматриваемом примере используется упрощенная модель системы автоматизации торговой деятельности. Имеются две таблицы, в одной из которых содержится список торговых представителей, а в другой - список клиентов. В процессе настройки производится репликация данных этих таблиц между одной консолидированной базой данных и одной удаленной базой данных. Для выполнения описываемых процедур необходим один компьютер с установленным ПО.

Первые шаги

Создание учетной записи

Для работы с данным учебным разделом необходимо обладать полномочиями системного администратора и полномочиями для работы с сервером Adaptive Server Enterprise. В этом учебном разделе предполагается, что именем для входа в систему является слово из двух символов – **sa**, а паролем - **sysadmin**.

В учебном разделе применяется утилита *isql* Adaptive Server Enterprise. Используя указанные выше имя и пароль, можно выполнить подключение к серверу Adaptive Server Enterprise с помощью следующей команды:

```
isql -S имя-сервера -U sa -P sysadmin,
```

где *имя-сервера* - имя сервера Adaptive Server Enterprise, к которому выполняется подключение.

Перед началом работы с учебным разделом убедитесь в наличии действительной учетной записи для подключения к серверу.

Создание базы данных

На сервере Adaptive Server Enterprise создайте базу данных **hq** с достаточным объемом для хранения таблиц и данных, необходимых для учебной базы данных.

Объем в 4 Мб будет достаточным.

❖ Процедура создания базы данных

- 1 Выполните подключение к серверу как пользователь с полномочиями системного администратора с помощью утилиты *isql*:

```
isql -S имя-сервера -U sa -P sysadmin
```

- 2 Активизируйте главную базу данных:

```
use master  
go
```

- 3 Создайте базу данных с именем **hq**. В этом примере рассматривается база данных 5 Мб с журналом 5 Мб на двух различных устройствах:

```
create database hq
on устройство_БД = 5
log on устройство_журнала = 5
go
```

☞ Для получения дополнительной информации о создании баз данных и выделения для них дискового пространства см. документацию по Adaptive Server Enterprise.

Установка SQL Remote

В базу данных **hq** необходимо установить функции SQL Remote.

❖ Процедура установки SQL Remote в базу данных hq

- 1 Если для используемого имени системного администратора база данных **hq** не является базой данных по умолчанию, создайте резервную копию сценария *ssremote.sql* из папки установки и добавьте в начало сценария следующие две строки:

```
use hq
go
```

- 2 Перейдите в папку учебного раздела. С помощью *isql* выполните подключение к серверу с использованием базы данных **hq** и запустите сценарий *ssremote.sql* из папки установки SQL Remote. Следующую команду необходимо ввести одной строкой:

```
isql -S имя-сервера -U sa -P sysadmin -i ssremote.sql
```

- 3 Если для используемого имени системного администратора база данных **hq** не является базой данных по умолчанию, создайте резервную копию сценария *stableq.sql* из папки установки и добавьте в начало сценария следующие две строки:

```
use hq
go
```

- 4 С помощью *isql* выполните подключение к серверу с использованием базы данных **hq** и запустите сценарий *stableq.sql* из папки установки SQL Remote. Следующую команду необходимо ввести одной строкой:

```
isql -S имя-сервера -U sa -P sysadmin -i stableq.sql
```

Создание папок для хранения сообщений

Создайте папку, в которой будут храниться файлы, используемые в данном учебном разделе. Например:

```
mkdir c:\tutorial
```

Необходимо создать папку для каждого из двух пользователей системы репликации под родительской папкой для данного учебного раздела:

```
mkdir c:\tutorial\hq
mkdir c:\tutorial\field
```

Следующим шагом является добавление пары таблиц в консолидированную базу данных.

❖ Процедура добавления таблиц в консолидированную базу данных

- 1 Выполните подключение к базе данных **hq** из *isql* в качестве системного администратора.
- 2 Активизируйте базу данных **hq**:

```
use hq
go
```

- 3 Создайте таблицу **SalesRep** при помощи следующего оператора:

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

- 4 Создайте таблицу **Customer** при помощи следующего оператора:

```
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 Измените таблицу **Customer** для добавления внешнего ключа в таблицу **SalesRep**:

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

Подготовка к работе с учебным разделом завершена.

Настройка консолидированной базы данных

В данном подразделе учебного раздела описывается процесс подготовки консолидированной базы данных простой системы репликации.

Процедура подготовки консолидированной базы данных для репликации включает следующие шаги:

- 1 Создание типа сообщений для использования при репликации.
- 2 Предоставление полномочий PUBLISH соответствующему пользователю для определения источника исходящих сообщений.
- 3 Предоставление полномочий REMOTE для всех пользователей-получателей сообщений.
- 4 Создание публикации с описанием реплицируемых данных.
- 5 Создание подписки с описанием предполагаемых получателей публикации.

Для выполнения этих задач требуется наличие полномочий системного администратора.

Создание систем передачи сообщений и адресов

В данном учебном разделе передача сообщений осуществляется через систему передачи сообщений путем совместного доступа к файлам. Необходимо создать тип сообщений FILE и указать адрес издателя консолидированной базы данных.

❖ Процедура создания типа сообщений

- ◆ Выполните хранимую процедуру **sp_remote_type**, используя HQ в качестве адреса издателя консолидированной базы данных:

```
sp_remote_type file, hq  
go
```

Адресом (**hq**) для системы передачи file является папка, в которой помещаются файлы, содержащие сообщения. Адрес указывается относительно переменной среды SQLRemote или записи в реестре. Поскольку это значение не было задано, укажите адрес относительно папки, из которой запускается Message Agent. Для обеспечения правильной

интерпретации адресов Message Agent необходимо запускать из папки учебного раздела.

☞ Для получения информации об установке значения переменной SQLRemote см. раздел "Установка параметров управления типами сообщений" на стр. 186.

После определения типа сообщений можно перейти к созданию необходимых пользователей.

Создание необходимых пользователей и полномочий

Обязательным условием для функционирования системы SQL Remote является наличие набора пользователей и полномочий. Для данного учебного раздела необходимо следующее:

- ◆ Удаленный пользователь или подписчик с именем **field_user**.
- ◆ Пользователь-издатель с именем **hq_user**.

В данном разделе описаны шаги, которые необходимо выполнить для создания каждого пользователя и назначения им необходимых полномочий.

❖ Процедура создания издателя

- 1 Добавьте учетную запись **hq_user** с базой данных по умолчанию **hq** и полномочиями системного администратора:

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

- 2 Добавьте учетную запись в базу данных HQ:

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

- 3 Назначьте этого пользователя издателем базы данных HQ:

```
exec sp_publisher hq_user
go
```

Добавление удаленного пользователя

Идентификация каждой удаленной базы данных в консолидированной БД производится по идентификатору пользователя с полномочиями REMOTE. Независимо от того, находится ли удаленная база данных на персональном сервере БД или на сетевом сервере с большим количеством пользователей, для ее идентификации в консолидированной базе данных необходим один идентификатор пользователя.

При настройке мобильной рабочей группы удаленные пользователи могут уже являться пользователями консолидированной базы данных, и в этом случае добавлять новых пользователей не требуется; однако возможно, что будет необходимо задать их статус как удаленных пользователей.

При добавлении к базе данных удаленного пользователя вместе с идентификатором этого пользователя необходимо указать используемую им систему передачи сообщений и адрес этой системы.

❖ Процедура создания подписчика

- 1 При отсутствии учетной записи для удаленного пользователя добавьте ее:

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

- 2 Добавьте пользователя в базу данных **hq**:

```
exec sp_adduser field_user
go
```

- 3 Предоставьте пользователю полномочия REMOTE. Выполните хранимую процедуру **sp_grant_remote**, используя имя пользователя **field_user**, тип сообщений **file** и соответствующую папку в качестве адреса:

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

Так же, как и адрес издателя, адрес удаленного пользователя (**field**) указывается относительно переменной среды SQLREMOTE или записи в реестре. Поскольку это значение не было задано, укажите адрес относительно папки, из которой запускается Message Agent. Для обеспечения правильной интерпретации адресов Message Agent необходимо запускать из папки учебного раздела.

☞ Для получения информации об установке значения переменной SQLRemote см. раздел "Установка параметров управления типами сообщений" на стр. 186.

Создание публикации и подписки

Последний шаг заключается в определении реплицируемых данных. Для этого необходимо сначала создать публикацию, в которой будут определены доступные данные, а затем создать подписку для **field_user**, в которой будут определены совместно используемые данные.

В Adaptive Server Enterprise такая публикация и подписка создаются при помощи процедуры **sp_create_publication** (создание пустой публикации) и процедуры **sp_add_article** (добавление статьи в процедуру). Также перед включением таблиц в публикацию необходимо отметить каждую таблицу как предназначенную для репликации.

❖ Процедура создания публикации

- 1 Создайте пустую публикацию:

```
exec sp_create_publication SalesRepData
go
```

- 2 Отметьте для публикации таблицу **SalesRep** и таблицу **Customer**:

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

- 3 Добавьте всю таблицу **SalesRep** в публикацию **SalesRepData**:

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

- 4 Добавьте таблицу **Customer** в публикацию **SalesRepData**, используя для разделения таблицы столбец **rep_key**. Следующий оператор необходимо ввести одной строкой, за исключением **go**:

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

Добавление подписки

Каждый пользователь, который будет получать информацию об изменениях в публикации, должен быть **подписан** на эту публикацию. Подписки могут создаваться только для действительных удаленных пользователей. В рассматриваемом примере требуется добавить подписку на публикацию **SalesRepData** для пользователя удаленной базы данных **field_user**.

❖ Процедура создания подписки

- 1 Создайте подписку в **SalesRepData** для **field_user** со значением подписки **rep1**:

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

На этом этапе подписка не **активизирована**, что означает, что обмена данными происходить не будет. Активизация подписки осуществляется с помощью утилиты извлечения базы данных.

Извлечение удаленной базы данных

Создание удаленной базы данных Adaptive Server Anywhere включает в себя три этапа:

- ◆ Извлечение структуры и данных в набор файлов. Это выполняется с помощью утилиты *ssextract*;
- ◆ Создание базы данных Adaptive Server Anywhere;
- ◆ Загрузка структуры и данных в базу данных.

Извлечение структуры и данных

Следующим шагом является извлечение базы данных Adaptive Server Anywhere со всей информацией для пользователя **field_user**. Данная процедура выполняется с помощью следующей команды (вводится одной строкой из папки учебного раздела):

```
ssextract -v -c "eng=имя-сервера;
dbn=hq;uid=sa;pwd=sysadmin" C:\tutorial\field field_user
```

Параметры имеют следующие значения.

- ◆ **-v** - расширенный режим. При разработке это обеспечивает вывод подробной информации.
- ◆ **-c** - параметр строки подключения. Строка подключения вводится в двойных кавычках после параметра **-c**.
- ◆ **eng=имя-сервера** - определяет сервер, к которому подключается утилита извлечения.
- ◆ **dbn=hq** - определяет используемую базу данных на сервере; в данном случае это база данных **hq**.
- ◆ **uid=sa** - имя пользователя для получения доступа к базе данных.
- ◆ **pwd=sysadmin** - пароль для получения доступа к базе данных.
- ◆ **C:\tutorial\field** - папка для размещения файлов, содержащих данные.
- ◆ **field_user** - идентификатор пользователя, для которого будет извлекаться база данных.

☞ Для получения подробной информации об утилите *dbxtract* и ее параметрах см. раздел "Утилита извлечения" на стр. 265.

В результате выполнения этой команды будут получены следующие файлы:

- ◆ **Сценарий перезагрузки.** Сценарий перезагрузки с именем *reload.sql* помещается в текущую папку.
- ◆ **Файлы данных.** Файлы, содержащие данные для загрузки в базу данных. В данном случае эти файлы будут пустыми.

Создание базы данных Adaptive Server Anywhere

Базу данных Adaptive Server Anywhere можно создать с помощью утилиты *dbinit*. В отличие от баз данных Adaptive Server Enterprise, простая база данных Adaptive Server Anywhere является файлом.

Необходимо создать такую базу данных Adaptive Server Anywhere, чтобы она была совместима с поведением, установленным для базы данных Adaptive Server Enterprise (за исключением случая, когда на сервере Adaptive Server Enterprise заданы параметры, отличные от параметров по умолчанию).

❖ **Процедура создания файла базы данных с именем field.db**

- ◆ Введите следующую команду из *c:\tutorial\field* directory:

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

Параметр `-b` активизирует режим дополнения пробелами в строковых сравнениях. Параметр `-c` активизирует режим учета регистра в строковых сравнениях. Параметр `-k` повышает степень совместимости системной папки с Adaptive Server Enterprise.

Загрузка данных в базу данных

Данные в базу данных можно загрузить с помощью утилиты Interactive SQL Adaptive Server Anywhere или утилиты *rtsql*. Последняя может использоваться как альтернатива Interactive SQL только для пакетной обработки и предназначена для работы в открытой базе данных.

❖ **Процедура загрузки данных в базу данных с использованием Interactive SQL**

- 1 Запустите сервер Adaptive Server Anywhere для базы данных **field**:

```
dbeng8 field.db
```

- 2 Выполните подключение к серверу с использованием утилиты Interactive SQL:

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

Необходимо ввести идентификатор пользователя и пароль (прописными буквами), поскольку база данных Adaptive Server Anywhere создана с учетом регистра.

- 3 Загрузите данные с помощью команды READ:

```
READ C:\TUTORIAL\RELOAD.SQL
```

❖ **Процедура пакетной загрузки данных в базу данных**

- 1 Запустите сервер Adaptive Server Anywhere для базы данных **field**:

```
dbeng8 field.db
```

- 2 Запустите сценарий из Interactive SQL:

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

Необходимо ввести идентификатор пользователя и пароль (прописными буквами), поскольку база данных Adaptive Server Anywhere создана с учетом регистра.

Что дальше?

Теперь система готова к репликации.

☞ Для перехода к следующему шагу – вставке и репликации данных – см. раздел "Запуск репликации данных" на стр. 42.

Запуск репликации данных

Теперь у пользователя имеется сконфигурированная система репликации. В данном разделе описывается процесс репликации данных из консолидированной базы данных в удаленную базу данных и из удаленной БД в консолидированную.

Ввод данных в консолидированную базу данных

В данном разделе описывается процесс ввода данных в таблицы **SalesRep** и **Customer** в консолидированной базе данных (Adaptive Server Enterprise) и репликация этих данных в базу данных Adaptive Server Anywhere.

❖ Процедура ввода данных в базу Adaptive Server Enterprise

- 1 Выполните подключение к серверу Adaptive Server Enterprise из *isql*:

```
isql -S имя-сервера -U sa -P sysadmin
```

- 2 Убедитесь, что используется база данных **hq**, и введите следующие строки:

```
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
```

Клиент **Ocean Sports** назначен пользователю **Field User**, а клиент **Sports Plus** – пользователю **Another User**. Необходимо подтвердить изменения, поскольку SQL Remote выполняет репликацию только после подтверждения изменений.

После ввода данных в консолидированную базу данных необходимо передать требуемые строки в базу данных Adaptive Server Anywhere.

Отправка данных из консолидированной базы данных

Для отправки строк в удаленную базу данных необходимо запустить Message Agent в консолидированной базе данных. Программа *ssremote* выполняет функцию Message Agent для Adaptive Server Enterprise.

❖ Процедура репликации данных из Adaptive Server Enterprise

- ◆ Для запуска Message Agent в консолидированной базе данных введите в командной строке следующий оператор (одной строкой):

```
ssremote -c "eng=имя-сервера;dbn=hq;uid=sa;pwd=sysadmin"
```

- 3 После отправки сообщений нажмите кнопку Shutdown в окне Message Agent для завершения работы программы.

Получение данных в удаленной базе данных

Для получения в удаленной БД оператора вставки данных необходимо запустить Message Agent (*dbremote*) в удаленной базе данных.

❖ Процедура получения данных в Adaptive Server Anywhere

- 1 При запущенном сервере базы данных получите данные с помощью Message Agent для Adaptive Server Anywhere:

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

☞ Для получения дополнительной информации о параметрах *dbremote* см. в раздел "Message Agent" на стр. 9.

- 2 После получения сообщений нажмите кнопку Shutdown в окне Message Agent для завершения работы программы.

Во время работы базы данных в окне Message Agent отображается информация о состоянии. При работе с реальной БД эту информацию можно поместить в файл журнала с целью ее сохранения.

Из записей видно, что сначала Message Agent получает сообщение из базы данных **hq**, а затем отправляет ответное сообщение. Возвращаемое сообщение содержит подтверждение успешного приема обновления при репликации; такие подтверждения являются частью системы отслеживания сообщений SQL Remote, обеспечивающей отправку сообщений даже в случае сбоев системы передачи сообщений.

Проверка получения данных

Теперь необходимо выполнить подключение к удаленной базе данных **field** из Interactive SQL и проверить, имеются ли полученные строки в таблицах **SalesRep** и **Customer**.

❖ Процедура проверки получения данных

- 1 Выполните подключение к базе данных **field** из Interactive SQL.
- 2 Просмотрите содержимое таблицы **SalesRep** путем выполнения следующего оператора:

```
SELECT * FROM SalesRep
```

Будет видно, что таблица **SalesRep** содержит обе строки, введенные в консолидированной базе данных. Это обусловлено тем, что в публикацию **SalesRepData** были включены все данные из таблицы **SalesRep**.

- 3 Просмотрите содержимое таблицы **Customer** путем выполнения следующего оператора:

```
SELECT * FROM Customer
```

Будет видно, что таблица **Customer** содержит только одну строку (**Ocean Sports**), введенную в консолидированной базе данных. Это происходит потому, что в публикации **SalesRepData** были включены только те клиенты, которые назначены указанному в подписке торговому представителю.

Репликация из удаленной базы данных в консолидированную базу данных

Теперь можно попробовать ввести данные в удаленной базе данных и затем передать эти данные в консолидированную БД. Далее представлены только основные шаги этой процедуры.

❖ Процедура репликации данных из удаленной базы данных в консолидированную базу данных

- 1 Выполните подключение к базе данных **field** из Interactive SQL.
- 2 Вставьте строку в удаленную базу данных (оператор INSERT). Например:

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

- 3 Подтвердите вставку строки (оператор COMMIT).
COMMIT;
- 4 При запущенной базе данных *field.db* выполните утилиту *dbremote* из командной строки для отправки сообщения в консолидированную базу данных.

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

- 5 Запустите *ssremote* для получения сообщения в консолидированной базе данных:

```
ssremote-c
"eng=имя-сервера;dbn=hq;uid=sa;pwd=sysadmin"
```

- 6 Выполните подключение к консолидированной базе данных и просмотрите содержимое таблицы **Customer**. Теперь в этой таблице три строки:

```
SELECT * FROM Customer
```

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

В этом простом примере средства предотвращения повторения значений первичного ключа не применяются. Однако в SQL Remote такие средства безопасности предусмотрены. Для получения подробной информации см. разделы, посвященные созданию схем репликации SQL Remote.

Создание схем репликации SQL Remote

В данной части рассматриваются некоторые аспекты создания схем репликации SQL Remote.



Принципы настройки SQL Remote

Об этой главе В данной главе описаны общие положения и принципы настройки инсталляции SQL Remote.

☞ Для получения подробной информации по различным системам см. главы "Настройка SQL Remote для Adaptive Server Enterprise" на стр. 121 и "Настройка SQL Remote для Adaptive Server Anywhere" на стр. 77.

Содержание

Раздел	Страница
Общие сведения о настройке	64
Как реплицируются операторы	67
Как реплицируются типы данных	71
Кто и что получает?	73
Ошибки и конфликты репликации	75

Общие сведения о настройке

В данной главе описываются основные положения проектирования публикаций, которые должны учитываться при настройке систем SQL Remote. Здесь также описан процесс репликации данных SQL Remote.

Проектирование в консолидированной базе данных

Подобно всем задачам по администрированию SQL Remote, проектирование также выполняется администратором базы данных или системным администратором в консолидированной базе данных.

Все задачи, связанные с конфигурированием SQL Remote, выполняются системным администратором Adaptive Server Enterprise или администратором базы данных.

Обеспечение совместимости баз данных

Необходимо обеспечить совместимость всех баз данных, задействованных в системе SQL Remote, в следующих категориях: порядок сортировки, кодовые таблицы и установка параметров баз данных.

Если в системе репликации используются базы данных Adaptive Server Enterprise и Adaptive Server Anywhere, необходимо проверить, что созданные базы данных Adaptive Server Anywhere совместимы с базами данных Adaptive Server Enterprise.

☞ Полное описание процесса создания баз данных Adaptive Server Anywhere, совместимых с Enterprise, см. в разделе "Создание баз данных, совместимых с Transact-SQL" (Creating a Transact-SQL-compatible database) на стр. 393 в документе "Руководство пользователя ASA SQL" (*ASA SQL User's Guide*). В данном разделе представлено только краткое описание.

❖ Процедура создания базы данных Adaptive Server Anywhere, совместимой с базой данных Enterprise (Sybase Central)

- ◆ В мастере создания базы данных (Create Database) имеется кнопка, с помощью которой можно выбрать требуемый режим эмуляции Adaptive Server Enterprise. Это самый простой способ создания базы данных, совместимой с Transact-SQL.

❖ Процедура создания базы данных Adaptive Server Anywhere (командная строка)

- 1 **Убедитесь, что конечные пробелы не учитываются.** Это выполняется при помощи параметра *dbinit -b*.
- 2 **Убедитесь, что задан идентификатор пользователя.** Если база данных уже имеет **идентификатор** пользователя с именем **dbo**, тогда владельцем системных представлений Transact-SQL Adaptive Server Anywhere можно назначить другого пользователя. Это выполняется с помощью параметра *dbinit -g*.
- 3 **Удалите старые системные представления.** Это выполняется с помощью параметра *dbinit -k*.
- 4 **Включите режим учета регистра в базе данных.** Это выполняется с помощью параметра *dbinit -c*.

Следующая команда создает в текущей папке чувствительную к регистру базу данных *test.db* с использованием текущего имени пользователя **dbo**, без учета конечных пробелов и с удалением старых системных представлений:

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


Использование совместимых порядков сортировки и кодовых таблиц

SQL Remote Message Agent не выполняет конвертации кодовых таблиц.

Наборы символов в системах Adaptive Server Anywhere

Для системы Adaptive Server Anywhere кодовая таблица и порядок сортировки, используемые в консолидированной базе данных, должны совпадать с таковыми в удаленной базе данных. Для получения информации о поддерживаемых кодовых таблицах см. раздел "Различные языки и кодовые таблицы" (International Languages and Character Sets) на стр. 249 в документе "Руководство по администрированию баз данных ASA" (ASA Database Administration Guide).

Наборы символов в системах Adaptive Server Enterprise

Библиотеки Open Client/Open Server выполняют конвертацию кодовых таблиц между SSREMOTE и Adaptive Server Enterprise всякий раз, когда кодовая таблица LOCALES.DAT отличается от кодовой таблицы Adaptive Server Enterprise. Обе кодовые таблицы должны быть установлены на сервере Adaptive Server Enterprise, также должна поддерживаться функция конвертации.

Наборы символов в смешанных системах

Параметры *locales.dat*, используемые всеми приложениями Open Client, должны соответствовать параметрам удаленного Adaptive Server Anywhere.

В таблице ниже представлены рекомендуемые соответствия между кодовыми таблицами Adaptive Server Anywhere и Adaptive Server Enterprise. Приведенная информация не является исчерпывающей.

Имя порядка сортировки в Adaptive Server Anywhere	Имя Open Client/Open Server	Порядок сортировки с учетом регистра в Open Client/Open Server	Порядок сортировки без учета регистра в Open Client/Open Server
значение по умолчанию	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

Имя порядка сортировки в Adaptive Server Anywhere	Имя Open Client/Open Server	Порядок сортировки с учетом регистра в Open Client/Open Server	Порядок сортировки без учета регистра в Open Client/Open Server
850CYR	cp850	bin_cp850	bin_cp850
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
EUC_KOREA	eucksc	bin_eucksc	bin_eucksc
UTF8	utf8	bin_utf8	bin_utf8

Как реплицируются операторы

Репликация SQL Remote основана на журнале транзакций, позволяющем при каждом обновлении реплицировать только изменения в данных, а не данные целиком. Фраза "SQL Remote реплицирует данные" на самом деле означает, что *"SQL Remote реплицирует операторы SQL, с помощью которых вносятся изменения в данные"*.

Реплицируются только подтвержденные транзакции	SQL Remote производит репликацию операторов только в подтвержденных транзакциях, что обеспечивает надлежащую атомарность транзакций в системе репликации и поддержку согласованности данных между базами данных, задействованных в репликации (с небольшой задержкой по времени при репликации данных).
Первичные ключи	<p>При репликации UPDATE или DELETE SQL Remote использует столбцы первичного ключа в целях однозначной идентификации обновляемой или удаляемой строки.</p> <p>Все реплицируемые таблицы должны иметь объявленный первичный ключ или ограничение уникальности. Одно уникального индекса недостаточно. Столбцы первичного ключа используются в разделе WHERE реплицированных обновлений или удалений. Если таблица не имеет первичного ключа, раздел WHERE относится ко всем столбцам в таблице.</p>
UPDATE - не всегда UPDATE	<p>При вводе простого оператора INSERT в одну базу данных он передается в другие базы данных в настроенной системе SQL Remote как оператор INSERT. Однако не все операторы реплицируются именно так, как они введены клиентским приложением. В данном разделе описывается репликация операторов SQL в SQL Remote. При необходимости создания надежной системы репликации SQL Remote изложенный здесь материал требует особого внимания.</p> <p>Компонентом, выполняющим репликацию операторов, является Message Agent.</p>

Репликация операторов вставки и удаления

Самый простой тип репликации применяется в отношении операторов INSERT и DELETE.

SQL Remote получает информацию по каждой операции INSERT или DELETE из журнала транзакций и передает ее во все узлы, подписавшиеся на операции вставки или удаления строки.

Если в подписке указано обновление только для определенного поднабора столбцов, передаваемые подписчикам операторы INSERT содержат только эти столбцы.

В Message Agent предусмотрено блокирование репликации операторов пользователю, который их изначально ввел.

Репликация операторов обновления

Операторы UPDATE реплицируются не совсем в том виде, в каком они вводятся из клиентского приложения. В данном разделе приведены два отличия реплицированного оператора UPDATE от введенного оператора UPDATE.

Репликация операторов UPDATE как операторов INSERT или DELETE	Если оператор UPDATE должен удалить строку, указанную в той или иной пользовательской подписке, он передается этому пользователю как оператор DELETE. Если оператор UPDATE должен добавить строку, указанную в той или иной
---------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

иной пользовательской подписке, он передается этому пользователю как оператор INSERT.

На рисунке показана публикация, где подписки представлены по имени подписчика.

Консолидированная БД			Ann		Marc	
ID	Rep	Dept	ID	Rep	ID	Rep
1	Ann	101	1	Ann	2	Marc
2	Marc	101			3	Marc
3	Marc	101				

Консолидированная БД			Ann		Marc	
ID	Rep	Dept	ID	Rep	ID	Rep
1	Ann	101	1	Ann	2	Marc
2	Marc	101	3	Ann	3	Marc
3	Ann	101				

Оператор UPDATE, изменяющий значение строки **Rep** с Marc на Ann, реплицируется пользователю Marc как оператор DELETE, а пользователю Ann – как оператор INSERT.

Такое переназначение строк среди подписчиков иногда называется **перераспределением областей**, потому что оно является обычным средством в приложениях автоматизации торговой деятельности, где торговые представители могут обмениваться клиентами.

Обнаружение конфликтов UPDATE

Оператор UPDATE изменяет значение одной или более строк с существующего на новое. Количество изменяемых строк указано в разделе WHERE оператора UPDATE.

Репликация оператора UPDATE в SQL Remote производится как обновление наборов отдельных строк. Эти операторы отдельных строк могут не выполняться по одной из следующих причин:

- ♦ **Строка для обновления не существует.** Каждая строка идентифицируется по ее значениям первичного ключа, и если первичный ключ был изменен другим пользователем, то строка, предназначенная для обновления, перестает существовать.

В этом случае оператор UPDATE ничего не обновляет.

- ♦ **Строка для обновления отличается по одному или нескольким столбцам.** Если другой пользователь изменил одно из значений, которые должны присутствовать в строке, возникает **конфликт обновления**.

В удаленных базах данных обновление осуществляется независимо от значений в строке.

Для консолидированной базы данных в SQL Remote обеспечивает операции **разрешения конфликтов**. Операции разрешения конфликтов содержатся в триггере или хранимой процедуре и запускаются автоматически при обнаружении конфликта.

В Adaptive Server Anywhere триггер разрешения конфликтов запускается перед обновлением, и само обновление выполняется по завершении триггера. В Adaptive Server Enterprise процедура разрешения конфликтов запускается после выполнения обновления.

- ♦ Таблица без первичного ключа или ограничения уникальности имеет ссылку на все столбцы в разделе WHERE реплицируемых операторов обновления.

Если два пользователя одновременно пытаются обновить одну и ту же строку, обновление согласно полученной при репликации информации не выполняется, вследствие чего базы данных утрачивают согласованность. Все реплицируемые таблицы должны иметь первичный ключ или ограничение уникальности; столбцы, указанные в ограничении, не должны обновляться.

Репликация процедур

Любая система репликации при репликации вызова хранимой процедуры сталкивается с выбором между двумя вариантами:

- ◆ **Репликация вызова процедуры.** Соответствующая процедура выполняется на узле репликации.
- ◆ **Репликация действий процедуры.** Репликация отдельных действий процедуры (INSERT, UPDATE, DELETE и т.д.).

SQL Remote реплицирует процедуры путем репликации действий процедуры.

Вызов процедуры не реплицируется.

Репликация триггеров

Репликация триггеров в SQL Remote выполняется по-разному в Message Agent для Adaptive Server Enterprise и Message Agent для Adaptive Server Anywhere.

Репликация триггеров из Adaptive Server Enterprise

Из Adaptive Server Enterprise реплицируются действия триггеров. Необходимо убедиться, что триггеры не запущены в удаленных базах данных Adaptive Server Anywhere. Если триггер запускается во время репликации, его действия будут выполняться дважды.

Параметр FIRE_TRIGGERS базы данных Adaptive Server Anywhere препятствует запуску триггеров. Если для идентификатора пользователя, используемого в Message Agent, этот параметр установлен, следите за тем, чтобы не использовать данный идентификатор пользователя для других целей.

Альтернативным подходом к предотвращению выполнения триггера, доступного только для Adaptive Server Anywhere, является использование следующего условия вокруг тела триггеров:

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

Это устанавливает зависимость выполнения триггера от того, является ли текущий пользователь Message Agent.

Репликация триггеров из Adaptive Server Anywhere

По умолчанию Message Agent для Adaptive Server Anywhere не реплицирует действия, выполняемые триггерами; предполагается, что триггер определен удаленно. Этим предотвращается предоставление полномочий и возможность повторного выполнения каждого действия. Впрочем, из этого правила есть несколько исключений:

- ◆ **Действия триггера по разрешению конфликтов.** Действия, выполняемые триггерами разрешения конфликтов (RESOLVE UPDATE), реплицируются из консолидированной базы данных во все удаленные базы данных, включая ту, из которой было послано сообщение, вызвавшее конфликт.
- ◆ **Репликация триггеров BEFORE.** Некоторые триггеры BEFORE могут привести к нежелательным результатам при использовании SQL Remote, поэтому действия триггера BEFORE по изменению обновляемой строки реплицируются прежде, чем будут выполнены действия UPDATE.

Это следует учитывать при настройке системы репликации. Например, BEFORE UPDATE, сталкивающийся со столбцом счетчика в строке для

отслеживания количества обновлений строки, при репликации проведет двойной подсчет, поскольку при репликации UPDATE выполнится триггер BEFORE UPDATE. Для предотвращения этой проблемы нужно убедиться в том, что в базе данных подписчика триггер отсутствует или не повлечет за собой повторное выполнение действий. При этом BEFORE UPDATE, который устанавливает столбец на время последнего обновления, также будет использовать время репликации UPDATE.

Параметр репликации действий триггеров

В Message Agent для Adaptive Server Anywhere имеется параметр, который вызывает репликацию всех действий триггеров при отправке сообщений. Это параметр *dbremote -t*.

При использовании этого параметра необходимо убедиться в том, что в удаленных базах данных действия триггеров не выполняются дважды: один раз - триггером, запущенным на удаленном узле, и второй – при явном выполнении действий, реплицированных из консолидированной базы данных.

Во избежание повторного выполнения действий триггеров можно обернуть оператор IF CURRENT REMOTE USER IS NULL ... END IF вокруг тела триггеров или установить для идентификатора пользователя Message Agent параметр Fire_triggers в Adaptive Server Anywhere в OFF.

Репликация операторов определения данных

Операторы определения данных (CREATE, ALTER, DROP и другие, изменяющие объекты базы данных) не реплицируются в SQL Remote, за исключением случаев ввода этих операторов в режиме ретрансляции.

☞ Для получения информации о режиме ретрансляции для Adaptive Server Anywhere см. раздел "Использование режима ретрансляции" на стр. 267.

Как реплицируются типы данных

Длинные двоичные или символьные данные, а также данные даты и времени требуют особого рассмотрения.

Репликация blob-объектов

Blob-объектами являются типы данных LONG VARCHAR, LONG BINARY, TEXT и IMAGE, значения которых превышают 256 символов.

Репликация Adaptive Server Anywhere

В SQL Remote предусмотрен особый метод репликации blob-объектов между базами данных Adaptive Server Anywhere.

Message Agent вместо значения использует в реплицируемом операторе INSERT или UPDATE переменную. Значение переменной строится при помощи последовательности операторов в следующей форме:

```
SET vble = vble || 'далее'
```

При этом размер операторов SQL с задействованными длинными значениями становится меньше, что позволяет поместить их в одно сообщение. Операторы SET являются отдельными операторами SQL, поэтому blob-объект эффективно разбивается по нескольким сообщениям SQL Remote.

Репликация Adaptive Server Enterprise

Blob-объекты можно реплицировать как в Adaptive Server Enterprise, так и из него, если эти объекты не превышают объем памяти Message Agent.

Приложения Sybase Open Client CTLIB, управляющие структурой CS_IODESC, не должны устанавливать параметр **log_on_update** в FALSE.

Использование параметра Verify_threshold для сведения к минимуму размера сообщения

Параметр **Verify_threshold** базы данных препятствует проверке правильности длинных значений (в разделе VERIFY реплицированного UPDATE). Значение по умолчанию для данного параметра - 1000. Если тип данных столбца длиннее порогового значения, тогда при репликации UPDATE все старые значения столбца не проверяются. Это сокращает размер сообщений SQL Remote, однако имеет свои недостатки: обнаружения конфликтов обновлений для длинных значений также не происходит.

Существует методика, позволяющая обнаруживать конфликты во время использования **Verify_threshold** для уменьшения размеров сообщений. Всякий раз при обновлении blob-объекта столбец **last_modified** в той же таблице также должен быть обновлен. После этого обнаружение конфликтов будет выполняться, потому что старое значение столбца **last_modified** подтверждено.

Использование рабочей таблицы во избежание избыточного обновления

Повторные обновления blob-объекта должны осуществляться в "рабочей" таблице, а окончательный вариант необходимо занести в реплицированную таблицу. Например, если рабочий документ обновляется 20 раз в течение дня, а Message Agent запущен только один раз в конце дня, тогда реплицированы будут 20 обновлений. Если длина документа - 200 Кб, тогда будет отправлено 4 Мб сообщений.

Лучшим решением в этой ситуации будет составление таблицы **document_in_progress**. По завершении редактирования документа пользователем приложение перемещает его (документ) из таблицы **document_in_progress** в реплицированную таблицу. Результатом становится одно обновление (200 Кб сообщений).

Управление репликацией blob-объектов

Параметр BLOB_THRESHOLD в Adaptive Server Anywhere позволяет осуществлять дальнейшее управление репликацией длинных значений. Любое значение длиннее параметра BLOB_THRESHOLD реплицируется как blob-объект. Это означает, что оно разбивается на части и реплицируется по частям, после чего

происходит его восстановление с помощью переменной SQL и сборка частей в узле получателя.

При задании высокого значения параметра BLOB_THRESHOLD в удаленных базах данных Adaptive Server Anywhere blob-объекты не разбиваются на части, и все операции можно применять к Adaptive Server Enterprise при помощи Message Agent. Операторы SQL не могут превосходить по размеру одно сообщение, поэтому в данном случае реплицировать можно только небольшие blob-объекты.

Репликация дат и времени

При репликации столбцов даты или времени для установки формата даты Message Agent использует установки параметров SR_Date_Format, SR_Time_Format и SR_Timestamp_Format базы данных.

Например, следующая установка параметра дает Message Agent команду отправить дату "2 мая 1987" в формате 1987-05-02.

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

☞ Для получения дополнительной информации см. раздел "Параметры SQL Remote" на стр. 272.

При репликации дат и времени рекомендуется учитывать следующее:

- ◆ Форматы времени, даты и метки времени должны быть согласованы в пределах всей системы.
- ◆ Если консолидированной базой данных является база данных Adaptive Server Anywhere, убедитесь, что порядок указания года, месяца и дня, используемый для форматов даты и метки времени, соответствуют установке параметра DATE_ORDER базы данных.

На время каждого подключения параметр DATE_ORDER можно изменять.

- ◆ Если консолидированной базой данных является база данных Adaptive Server Enterprise, убедитесь, что порядок указания года, месяца и дня в установках SQL Remote соответствует установке *dateformat* в базе данных Adaptive Server Enterprise.

❖ Процедура поиска установок формата даты в базе данных Adaptive Server Enterprise

- 1 Войдите в базу данных Adaptive Server Enterprise из *isql* под именем пользователя, который используется *ssremote*. В данном примере в качестве имени пользователя используется *ssr*.
- 2 Введите следующую команду:

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

Adaptive Server Enterprise возвращает язык по умолчанию для пользователя *ssr*.

- 3 Если *ssr* использует язык по умолчанию (*us_english*), тогда формат даты по умолчанию - YMD. Если язык отличается от языка по умолчанию, введите следующую команду:

```
sp_helplanguage имя-языка,
```

где *имя-языка* – язык пользователя *ssr*. В отображаемой информации содержатся данные о формате даты по умолчанию для этого языка.

Кто и что получает?

Каждый раз при вставке, удалении или обновлении строки в таблице подписчикам на эту строку должно быть отправлено сообщение. Кроме этого, обновление может изменить выражение подписки, поэтому одним подписчикам отправляется оператор DELETE (удаление), другим – UPDATE (обновление), а третьим – INSERT (вставка).

☞ Для получения подробной информации об операторах, отправляемых подписчикам, см. раздел "Как реплицируются операторы" на стр. 67. Для получения подробной информации о подписке см. следующие две главы.

В данном разделе описан процесс отправки SQL Remote требуемых операторов соответствующим получателям.

Задача определения того, кому и что необходимо получить, разделена между сервером базы данных и Message Agent. Процессор обрабатывает аспекты, имеющие отношение к публикациям, а Message Agent - к подпискам.

Действия Adaptive Server Anywhere

Adaptive Server Anywhere оценивает выражение подписки для каждого обновления таблицы, являющейся частью публикации. Он добавляет значение выражения в журнал как до, так и после обновления.

Список подписчиков не изменяется

Adaptive Server Enterprise не оценивает и не вводит в журнал список подписчиков. Оценивается и вводится выражение подписки (свойство публикации). Обработкой подписчиков занимается Message Agent.

Для таблицы, являющейся частью более одной публикации, выражение подписки оценивается до и после обновления каждой публикации.

Добавление информации в журнал может повлиять на производительность в следующих случаях:

- ◆ **Выражения с высокими затратами на выполнение.** Если оценить выражение подписки слишком сложно, это может повлиять на производительность.
- ◆ **Большое количество публикаций.** Если таблица принадлежит большому количеству публикаций, необходимо провести оценку большого количества выражений. Однако количество *подписок* не существенно.
- ◆ **Выражения с несколькими значениями.** Некоторые выражения многозначны. Это может привести к большому количеству дополнений при формировании журнала транзакций, что, соответственно, повлияет на производительность.

Действия Adaptive Server Enterprise

В SQL Remote для публикации Adaptive Server Enterprise выражение подписки должно быть столбцом. Столбец подписки содержит одиночное значение или список значений, разделенных запятыми.

Не список подписчиков

Adaptive Server Enterprise не вносит в журнал список подписчиков. Вводится только значение столбца. Обработкой подписчиков занимается Message Agent.

Если таблица отмечена для репликации с использованием `sp_add_remote_table` (который вызывает `sp_setreplicate`), Adaptive Server Enterprise размещает в журнале транзакций образ всей строки до обновления для удаления, образ всей строки после обновления для вставки, и оба эти образа для обновлений. Это означает, что будут доступны значения столбца подписки до и после обновления.

Действия Message Agent

Message Agent считывает оцененные выражения подписки или записи столбца подписки из журнала транзакций и сравнивает значения до и после обновления со значением подписки для каждого подписчика на публикацию. Таким образом

Message Agent может передавать корректные данные об операциях каждому подписчику.

Наличие большого количества подписчиков не влияет на производительность сервера, однако влияет на производительность Message Agent. Высокие затраты производительности могут быть вызваны как действиями по сопоставлению значений подписки с большим количеством этих значений, так и операциями передачи сообщений.

Ошибки и конфликты репликации

В SQL Remote обеспечивается возможность обновления баз данных на множестве различных узлов. Во избежание ошибок репликации проектирование необходимо проводить достаточно осторожно, особенно если база данных имеет сложную структуру. В данном разделе описываются типы ошибок и конфликтов, которые могут иметь место при настройке репликации; в последующих разделах описывается процесс построения публикаций без ошибок, а также способы разрешения конфликтов.

Ошибки доставки в данном разделе не рассматриваются

В данном разделе не рассматриваются ошибки, связанные со сбоями в передаче сообщений. Для получения информации об ошибках доставки и их устранении см. раздел "Система отслеживания сообщений" на стр. 205.

Ошибки репликации

Ошибки репликации можно разделить на следующие категории:

- ◆ **Ошибки дублирования первичного ключа.** Два пользователя выполняют вставку строки (INSERT) с одним и тем же первичным ключом, либо один пользователь обновляет первичный ключ, а другой вставляет первичный ключ с таким же значением. Попытка обращения к базе данных при выполнении второй операции будет в системе репликации неудачной, поскольку ее выполнение привело бы к дублированию первичного ключа.
- ◆ **Ошибки типа "Строка не найдена".** Пользователь удаляет (DELETE) строку (т.е. строку с данным значением первичного ключа). Второй пользователь выполняет обновление (UPDATE) или удаление (DELETE) той же строки на другом узле.

В этом случае второй оператор не сработает, поскольку строка не будет найдена.

- ◆ **Ошибки ссылочной целостности.** Если содержащий внешний ключ столбец включен в публикацию, а связанный с ним первичный ключ в публикацию не включен, тогда утилита извлечения не передает определение внешнего ключа из удаленной базы данных для обеспечения возможности выполнения оператора INSERT на удаленной базе данных.

Данная проблема решается вводом в определения таблицы всех необходимых значений по умолчанию.

Также ошибки ссылочной целостности могут возникать в случае, когда в первичной таблице имеется выражение SUBSCRIBED BY, а в связанной с ней внешней таблице такое выражение отсутствует: при этом строки из внешней таблицы могут быть реплицированы, однако строки из первичной таблицы могут быть исключены из публикации.

Конфликты репликации

Конфликты репликации по своей сути отличаются от ошибок. При условии надлежащей обработки конфликты не вызывают проблем в работе SQL Remote.

- ◆ **Конфликты.** Пользователь обновляет строку. Второй пользователь обновляет ту же строку на другом узле. Операция второго пользователя успешна, и SQL Remote позволяет запустить триггер (Adaptive Server Anywhere) или вызвать процедуру (Adaptive Server Enterprise) для

разрешения этих конфликтов способом, который является применимым для изменяемых данных.

Возникновение конфликтов происходит во многих системах репликации. Корректное разрешение конфликтов при помощи триггеров и процедур является одним из направлений нормальной работы SQL Remote.

☞ Информация об обработке SQL Remote конфликтов по мере их возникновения представлена в следующих главах.

Отслеживание ошибок SQL

При настройке системы ошибки SQL при репликации необходимо исключить. В SQL Remote предусмотрен параметр, который помогает отслеживать ошибки в операторах SQL, однако этот параметр не может использоваться для их устранения.

При установке параметра **Replication_error** можно указать хранимую процедуру, вызываемую Message Agent при возникновении ошибки SQL. По умолчанию вызова процедуры не происходит.

❖ Процедура установки параметра Replication_error в Adaptive Server Anywhere

- ◆ Введите следующий оператор:

```
SET OPTION
    удаленный-пользователь.Replication_error
    = 'имя-процедуры'
```

где *удаленный-пользователь* - **идентификатор** пользователя в командной строке Message Agent, а *имя-процедуры* – процедура, вызываемая при обнаружении ошибки SQL.

❖ Процедура установки параметра Replication_error в Adaptive Server Enterprise

- ◆ Введите следующий оператор:

```
exec sp_remote_option Replication_error, имя-процедуры
go
```

где *имя-процедуры* - процедура, вызываемая при обнаружении ошибки SQL.

Требования к
процедуре
Replication_error

Процедура Replication_error должна иметь один аргумент типа CHAR, VARCHAR или LONG VARCHAR. Данная процедура вызывается один раз при получении сообщения об ошибке SQL и один раз при выполнении оператора SQL, вызывающего ошибку.

Настройка SQL Remote для Adaptive Server Anywhere

Об этой главе

В данной главе описывается настройка инсталляции SQL Remote в случае, когда в качестве консолидированной базой данных используется база данных Adaptive Server Anywhere.

Схожий материал для Adaptive Server Enterprise

Многие принципы проектирования публикаций относятся в равной степени к Adaptive Server Anywhere и Adaptive Server Enterprise, но в отношении команд и возможностей существуют различия. Данная глава во многом совпадает с соответствующей главой, предназначенной для пользователей Adaptive Server Enterprise, "**Настройка SQL Remote для Adaptive Server Enterprise**" на стр. 121

Содержание

Раздел	Страница
Общие сведения о настройке	78
Публикация данных	79
Проектирование публикаций для Adaptive Server Anywhere	87
Разделение таблиц, не содержащих выражение подписки	90
Совместное использование строк в нескольких подписках	96
Разрешение конфликтов	102
Обеспечение уникальности первичных ключей	110
Создание подписок	118

Общие сведения о настройке

При настройке SQL Remote необходимо решить следующие задачи:

- ◆ **Проектирование публикаций.** Публикации определяют, какая информация совместно используется какими базами данных.
- ◆ **Проектирование подписок.** Подписки определяют, какую информацию получает каждый пользователь.
- ◆ **Реализация проекта.** Создание публикаций и подписок для всех пользователей в системе.

Все функции администрирования выполняются в консолидированной базе данных

Как и все задачи по администрированию SQL Remote, проектирование выполняется администратором базы данных или системным администратором в консолидированной базе данных.

Администратор базы данных Adaptive Server Anywhere должен выполнять все задачи по конфигурированию SQL Remote.

Публикация данных

В данном разделе описывается создание простых публикаций, состоящих из целых таблиц или из постолбцовых подмножеств таблиц; эти таблицы также называются статьями. Эти задания можно выполнить с помощью Sybase Central или используя оператор CREATE PUBLICATION в Interaction SQL.

Все публикации в Sybase Central появляются в папке Publications. Все статьи, создаваемые для заданной публикации, появляются в контейнере публикации.

Каждая публикация может содержать одну или более целых таблиц, но допускается также использование неполных таблиц. Таблица может быть подразделена на столбцы и строки.

Публикация целых таблиц

Самая простая публикация, которую можно создать, состоит из одной статьи, которая включает в себя все строки и столбцы одной или более таблиц. Эти таблицы должны быть уже созданы.

❖ Процедура публикации одной или более целых таблиц (Sybase Central)

- 1 Выполните подключение к базе данных как пользователь с полномочиями администратора БД.
- 2 Откройте папку Publications и дважды щелкните по пункту Add Publication.
- 3 Введите имя новой публикации. Нажмите кнопку Next.
- 4 На закладке Tables выберите таблицу из списка Matching tables. Нажмите кнопку Add. Выбранная таблица появится в списке Selected Tables справа.
- 5 При необходимости можно добавить дополнительные таблицы. Порядок добавления таблиц не имеет значения.
- 6 Нажмите кнопку Finish.

❖ Процедура публикации одной или более целых таблиц (SQL)

- 1 Выполните подключение к базе данных как пользователь с полномочиями администратора БД.
- 2 Выполните оператор CREATE PUBLICATION, который задает имя новой публикации и определяет таблицу, которую требуется опубликовать.

Пример

- ◆ С помощью приведенного ниже оператора создается публикация, в которой указана вся таблица *customer*:

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

- ◆ Следующий оператор создает публикацию, включающую все столбцы и строки из каждой таблицы демонстрационной базы данных Adaptive Server Anywhere:

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

☞ Для получения дополнительной информации см. раздел “Оператор CREATE PUBLICATION” (CREATE PUBLICATION statement) на стр. 314 в документе “Справочник по SQL для ASA” (ASA SQL Reference Manual).

Публикация только некоторых столбцов таблицы

Можно создать публикацию, которая содержит все строки и только некоторые столбцы таблицы. Это выполняется из Sybase Central или путем перечисления всех столбцов в операторе CREATE PUBLICATION.

❖ Процедура публикации только некоторых столбцов таблицы (Sybase Central)

- 1 Выполните подключение к базе данных как пользователь с полномочиями администратора БД.
- 2 Откройте папку Publications и дважды щелкните по пункту Add Publication.
- 3 Введите имя новой публикации. Нажмите кнопку Next.
- 4 На закладке Tables выберите таблицу из списка Matching tables. Нажмите кнопку Add. Выбранная таблица будет добавлена в список Selected Tables справа.
- 5 На закладке Columns дважды щелкните на значке таблицы, чтобы развернуть список доступных столбцов. Выберите каждый столбец, который необходимо опубликовать, и нажмите Add. Выбранные столбцы появятся в правой части окна.
- 6 Нажмите Finish.

❖ Процедура публикации только некоторых столбцов таблицы (SQL)

- 1 Выполните подключение к базе данных как пользователь с полномочиями администратора БД.
 - 2 Выполните оператор CREATE PUBLICATION, который определяет имя публикации и имя таблицы. Перечислите опубликованные столбцы в круглых скобках после имени таблицы.
- ◆ С помощью следующего оператора создается публикация, в которой указаны все строки столбцов *id*, *company_name* и *city* таблицы *customer*:

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

☞ Для получения дополнительной информации см. раздел “Оператор CREATE PUBLICATION” (CREATE PUBLICATION statement) на стр. 314 в документе “Справочник по SQL для ASA” (ASA SQL Reference Manual).

Публикация только некоторых строк таблицы

Можно создать публикацию, которая будет содержать все столбцы и только некоторые строки таблицы, из Sybase Central. В обоих случаях необходимо написать условие поиска, которое соответствует только тем строкам, которые требуется опубликовать.

Sybase Central и язык SQL предоставляют два способа публикации только некоторых строк таблицы, но только один из этих способов совместим с MobiLink.

- ◆ **Раздел WHERE.** Для включения поднабора строк в статью можно использовать раздел WHERE. Все подписчики на публикацию, содержащую эту статью, получают строки, которые удовлетворяют условию раздела WHERE.
- ◆ **Выражение подписки.** Выражение подписки можно использовать для включения различного набора строк в разные подписки на публикации, содержащие данную статью.

В статье можно объединять раздел WHERE и выражение подписки. Определить их можно в Sybase Central или в операторе CREATE PUBLICATION.

Выражение подписки используется тогда, когда разные подписчики на публикацию должны получать разные строки из таблицы. Выражение подписки является самым мощным инструментом разделения таблиц.

Раздел WHERE используется для исключения одного и тот же набора строк из всех подписок на публикацию.

Публикация только некоторых строк при помощи раздела WHERE

Можно определить раздел WHERE, чтобы включить в публикацию только те строки, которые отвечают условиям WHERE.

❖ Процедура создания публикации с помощью раздела WHERE (Sybase Central)

- 1 Выполните подключение к базе данных как пользователь с полномочиями администратора БД.
- 2 Откройте папку Publications и запустите мастер добавления публикации (Add Publication).
- 3 Введите имя новой публикации. Нажмите кнопку Next.
- 4 На закладке Tables выберите таблицу из списка Matching tables. Нажмите кнопку Add. Выбранная таблица будет добавлена в список Selected Tables справа.
- 5 На закладке Where выберите таблицу, после чего в нижнем поле введите условие поиска. Для установки формата условия поиска можно воспользоваться диалогом Insert.
- 6 Нажмите кнопку Finish.

❖ Процедура создания публикации с помощью раздела WHERE (SQL)

- 1 Выполните подключение к базе данных как пользователь с полномочиями администратора БД.
- 2 Выполните оператор CREATE PUBLICATION, который содержит строки, необходимые для включения в публикацию, и условие WHERE.
- ◆ При помощи следующего оператора создается публикация, в которой указаны столбцы id, company_name, city и state таблицы customer для клиентов, отмеченных как активные в столбце status.

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

Примеры

```
        city,  
        state )  
    WHERE status = 'active'  
    )
```

В данном случае столбец *status* не публикуется. Все неопубликованные строки должны иметь значение по умолчанию. В противном случае происходит ошибка при загрузке строк для вставки из консолидированной базы данных.

- ◆ Ниже приведена публикация единственной статьи, в которой торговому представителю Сэмюэлю Сингеру (Samuel Singer) посылается необходимая информация о заказе:

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

☞ Для получения дополнительной информации см. раздел “Оператор CREATE PUBLICATION” (CREATE PUBLICATION statement) на стр. 314 в документе “Справочник по SQL для ASA” (ASA SQL Reference Manual).

Раздел SUBSCRIBE BY

Оператор создания публикации также допускает использование раздела SUBSCRIBE BY. Данный раздел может также использоваться для выборочной публикации строк в SQL Remote. Однако он игнорируется при синхронизации MobiLink.

Публикация только некоторых строк при помощи выражения подписки

Можно определить выражение подписки, чтобы включить другой набор строк в различные подписки к публикациям, содержащие данную статью.

Например, в ситуации с интенсивно работающими мобильными приложениями публикация объема продаж может потребоваться тогда, когда каждый торговый представитель подписывается на свои собственные заказы на закупку, что позволяет им обновлять данные по своим заказам на местах и отправлять информацию по продажам в консолидированную базу данных.

При использовании модели раздела WHERE для каждого торгового представителя будет необходима отдельная публикация. Приводимая ниже публикация предназначена для торгового представителя Сэмюэля Сингера; такая публикация будет необходима для каждого торгового представителя.

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

Если в данной системе репликации требуется создать большое количество разных подписок, в SQL Remote допускается связывать **выражение подписки** со статьей. Количество получаемых подписками строк зависит от значения, указанного в выражении.

Преимущества использования выражений подписки

Публикации, в которых используется выражение подписки, более компактны и проще для понимания; они обеспечивают большую производительность, нежели поддержание нескольких публикаций раздела WHERE. Сервер базы данных должен добавлять информацию в журнал транзакций и сканировать журнал транзакций для отсылки сообщений, число которых прямо пропорционально количеству публикаций. Выражение подписки позволяет связывать большое количество различных подписок с отдельной публикацией, тогда как раздел WHERE этого не допускает.

❖ **Процедура создания статьи с помощью выражения подписки (Sybase Central)**

- 1 Выполните подключение к базе данных как пользователь с полномочиями администратора БД.
- 2 Откройте папку Publications (эта папка находится в папке SQL Remote).
- 3 Дважды щелкните по пункту Add Publication.
- 4 На первой странице мастера создания публикации (Publication Creation) введите имя публикации и нажмите кнопку Next.
- 5 На закладке Table установите необходимые значения для данной таблицы.
- 6 На закладке Subscribe by с помощью элементов управления создайте выражение подписки.
- 7 Выполните остальные указания мастера.

❖ **Процедура создания статьи с помощью выражения подписки (SQL)**

- 1 Выполните подключение к базе данных как пользователь с полномочиями администратора БД.
- 2 Выполните оператор CREATE PUBLICATION, который включает в себя выражение, используемое для сопоставления в выражении подписки.
- ◆ С помощью приведенного ниже оператора создается публикация, в которой указаны столбцы *id*, *company_name*, *city* и *state* таблицы *customer* и которая сопоставляет строки с подписками по значению столбца *state*:

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

- ◆ С помощью следующего оператора на публикацию подписываются два сотрудника: Энн Тэйлор (Ann Taylor) получает клиентов в Джорджии (GA), а Сэм Сингер (Sam Singer) - клиентов в Массачусетсе (MA).

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

CREATE SUBSCRIPTION
TO pub_customer ('MA')
FOR Sam_Singer
```

Пользователи могут подписаться как на одну, так и на большее число публикаций, а также иметь более одной подписки на отдельную публикацию.

☞ Для получения дополнительной информации см. следующие разделы:

- ◆ "Оператор CREATE PUBLICATION" (CREATE PUBLICATION statement) на стр. 314 в документе *"Справочник по SQL для ASA" (ASA SQL Reference Manual)*;
- ◆ "Разделение таблиц, не содержащих выражение подписки" на стр. 90;
- ◆ "Создание подписок" на стр. 118;
- ◆ "Публикация только некоторых строк при помощи раздела WHERE" на стр. 93;
- ◆ "Изменение существующих публикаций" на стр. 84.

Примеры

Изменение существующих публикаций

Изменить публикацию после ее создания можно путем добавления, изменения или удаления статей, или же посредством переименования самой публикации. Если в статью были внесены изменения, необходимо ввести полное описание измененной статьи.

Выполнить данные задачи можно при помощи Sybase Central или оператора ALTER PUBLICATION в Interactive SQL.

❖ Процедура внесения изменений в свойства существующих публикаций или статей (Sybase Central)

- 1 Выполните подключение к базе данных как владелец публикации или как пользователь с полномочиями администратора БД.
- 2 Щелкните правой кнопкой мыши по публикации или статье и выберите Properties во всплывающем меню.
- 3 Установите требуемые свойства.

❖ Процедура добавления статей (Sybase Central)

- 1 Выполните подключение к базе данных как владелец публикации или как пользователь с полномочиями администратора БД.
- 2 Откройте папку Publications (эта папка находится в папке SQL Remote).
- 3 Откройте контейнер публикации.
- 4 Дважды щелкните по пункту Add Article.
- 5 В мастере создания новой статьи (Create a New Article) выполните следующее:
 - ◆ Выберите таблицу и нажмите кнопку Next.
 - ◆ Выберите число столбцов. Нажмите кнопку Next.
 - ◆ Введите раздел WHERE (если требуется). Нажмите кнопку Next.
 - ◆ Создайте выражение подписки (если требуется).
- 6 Нажмите ОК для завершения создания статьи.

❖ Процедура удаления статей (Sybase Central)

- 1 Выполните подключение к базе данных как владелец публикации или как пользователь с полномочиями администратора БД.
- 2 Откройте папку Publications (эта папка находится в папке SQL Remote).
- 3 Откройте контейнер публикации.
- 4 Щелкните правой кнопкой мыши по статье, которую требуется удалить, и выберите Delete во всплывающем меню.

❖ Процедура изменения существующей публикации (SQL)

- 1 Выполните подключение к базе данных как владелец публикации или как пользователь с полномочиями администратора БД.
- 2 Выполните подключение к базе данных как пользователь с полномочиями администратора БД.
- 3 Выполните оператор ALTER PUBLICATION.

Пример

- ◆ С помощью приведенного ниже оператора таблица *customer* добавляется к публикации *pub_contact*.

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

☞ Для получения дополнительной информации см. следующие разделы:

- ◆ "Оператор ALTER PUBLICATION" (ALTER PUBLICATION statement) на стр. 216 в документе *"Справочник по SQL для ASA" (ASA SQL Reference Manual)*;
- ◆ "Публикация только некоторых строк при помощи раздела WHERE" на стр. 93;
- ◆ "Публикация только некоторых строк при помощи выражения подписки" на стр. 95.

Удаление публикаций

Удалить публикацию можно с помощью либо Sybase Central, либо оператора DROP PUBLICATION. При удалении публикации все подписки на данную публикацию автоматически удаляются.

Для удаления публикации необходимо обладать полномочиями администратора БД.

❖ Процедура удаления публикации (Sybase Central)

- 1 Выполните подключение к базе данных как пользователь с полномочиями администратора БД.
- 2 Откройте папку Publications.
- 3 Щелкните правой кнопкой мыши по требуемой публикации и выберите Delete во всплывающем меню.

❖ Процедура удаления публикации (SQL)

- 1 Выполните подключение к базе данных как пользователь с полномочиями администратора БД.
- 2 Выполните оператор DROP PUBLICATION.

Пример

С помощью приведенного ниже оператора удаляется публикация *pub_orders*.

```
DROP PUBLICATION pub_orders
```

☞ Для получения дополнительной информации см. раздел "Оператор DROP PUBLICATION" (DROP PUBLICATION statement) на стр. 402 в документе *"Справочник по SQL для ASA" (ASA SQL Reference Manual)*.

Примечания относительно публикаций

- ◆ Различные типы описанных выше публикаций можно объединять. С помощью отдельной публикации можно опубликовывать поднабор столбцов из набора таблиц и использовать раздел WHERE для выбора набора реплицируемых строк.
- ◆ Для создания и удаления публикаций необходимо иметь полномочия администратора БД.

- ◆ Публикации могут быть изменены только администратором БД или владельцем публикации.
- ◆ Изменение публикаций во время работы настроенного SQL Remote может вызвать ошибку репликации и привести к потере данных в системе репликации, поэтому данную процедуру необходимо выполнять с особой осторожностью.
- ◆ В публикации не могут быть включены представления.
- ◆ В публикации не могут быть включены хранимые процедуры. Для получения подробной информации о репликации процедур и триггеров в SQL Remote см. раздел "Репликация процедур" на стр. 69.

Проектирование публикаций для Adaptive Server Anywhere

Ознакомившись с процедурой создания простых публикаций, рассмотрим вопрос правильного проектирования публикаций. Правильное проектирование имеет немаловажное значение для создания системы SQL Remote. В данном разделе изложены принципы правильного проектирования применительно к SQL Remote для Adaptive Server Anywhere.

Схожий материал для Adaptive Server Enterprise

Многие принципы проектирования публикаций относятся в равной степени к Adaptive Server Anywhere и Adaptive Server Enterprise, но в отношении команд и возможностей существуют различия. Данная глава во многом совпадает с соответствующей главой, предназначенной для пользователей Adaptive Server Enterprise, "Проектирование публикаций для Adaptive Server Enterprise" на стр. 127.

Принципы настройки

Каждая подписка должна быть полной реляционной базой данных

Удаленная база данных использует информацию в своих подписках совместно с консолидированной базой данных. Подписка одновременно является поднабором реляционной базы данных, хранящейся на консолидированном узле, и полной реляционной базой данных на удаленном узле. Поэтому информация в подписке подчиняется тем же правилам, что и любая другая реляционная база данных:

- ◆ **Должны быть действительны связи по внешнему ключу.** Для каждой записи во внешнем ключе должна существовать соответствующая запись первичного ключа в базе данных.

Утилита извлечения базы данных производит проверку того, что операторы CREATE TABLE для удаленных баз данных не имеют внешних ключей, определенных для не существующих удаленно таблиц.

- ◆ **Должна сохраняться уникальность первичного ключа.** Невозможно проверить то, какие новые строки были введены на других узлах, но еще не были реплицированы. При проектировании должна быть исключена возможность добавления пользователями на различных узлах строк с идентичными значениями первичного ключа, поскольку это может привести к конфликтам при репликации строк в консолидированную базу данных.

Целостность транзакций при отсутствии блокирования должна сохраняться

Данные в рассредоточенной базе данных (которая состоит из консолидированной базы данных и всех удаленных баз данных) должны сохранять свою целостность при выполнении обновлений на всех узлах, даже если не существует механизма блокирования в масштабе всей системы для любой отдельной строки.

- ◆ **Должны предотвращаться или разрешаться конфликты блокирования.** В системе SQL Remote не существует способа блокирования строк во всех базах данных для предотвращения одновременного изменения строк различными пользователями. Подобные конфликты необходимо предотвращать путем выработки соответствующих решений за пределами системы, либо решать их должным образом в консолидированной базе данных.

Эти основные особенности реляционных баз данных должны быть учтены при разработке публикаций и подписок. В данной главе рассматриваются принципы и методы правильного проектирования.

Условия создания действительных статей

Поддержка операторов INSERT в удаленных базах данных

В статью должны быть включены все столбцы в первичном ключе.

Чтобы операторы INSERT в удаленной базе данных могли создавать точные копии в консолидированной базе данных, можно исключать из статьи только те столбцы, которые можно не включать в действующий оператор INSERT. Таковыми являются:

- ◆ Столбцы, которые допускают значение NULL;
- ◆ Столбцы, которые имеют значения по умолчанию.

Если исключить какой-либо столбец, не отвечающий одному из этих требований, операторы INSERT, выполняемые в удаленной базе данных, не будут работать после их репликации в консолидированную базу данных.

Консолидированная база данных

INSERT INTO SalesRep (ID, Rep) VALUES (3, 'Shih')

ID	Rep	Dept
1	Ann	101
2	Marc	101
3	Shih	X

Оператор Insert не выполняется

Удаленная база данных

INSERT INTO SalesRep (ID, Rep) VALUES (3, 'Shih')

ID	Rep
1	Ann
2	Marc
3	Shih

Оператор Insert успешно выполняется

Использование триггеров BEFORE в качестве альтернативы

Исключением в данном случае является использование базы данных Adaptive Server Anywhere в качестве консолидированной базы данных, однако при этом для поддержания столбцов, не включенных в оператор INSERT, необходимо вписать триггер BEFORE.

Советы по проектированию для улучшения производительности

В данном разделе приводится список рекомендаций по высокоэффективной настройке системы SQL Remote.

- ◆ **Не создавайте большое количество публикаций.** В частности, старайтесь не ссылаться на одну и ту же таблицу во многих различных публикациях.

Степень загруженности сервера базы данных пропорциональна количеству публикаций. Путем ввода небольшого числа публикаций и эффективного использования подписок можно снизить нагрузку на сервер базы данных.

При выполнении операций с таблицей сервер базы данных и Message Agent должны проделать определенную работу с каждой публикацией, содержащей таблицу. Наличие одной публикации для каждого удаленного пользователя чрезмерно увеличит нагрузку на сервер базы данных. Гораздо эффективнее создать несколько публикаций, использующих оператор SUBSCRIBE BY, и иметь подписки для каждого удаленного пользователя. Сервер базы данных не выполняет никаких дополнительных операций при добавлении к публикации дополнительных подписок. Message Agent спроектирован для эффективной работы с большим количеством подписок.

- ◆ **Группируйте публикации логично.** Например, если есть таблица, которая требуется каждому удаленному пользователю, к примеру, таблица прейскуранта, следует создать отдельную публикацию для этой таблицы. Создавайте по одной публикации на каждую таблицу, в которой данные могут быть разделены по значению столбца.
- ◆ **Эффективно применяйте подписки.** Когда удаленные пользователи получают схожие поднаборы консолидированной базы данных, всегда используйте публикации с выражением SUBSCRIBE BY. Не создавайте отдельную публикацию для каждого удаленного пользователя.
- ◆ **Обращайте особое внимание на триггеры обновления публикации (Update Publication).** В частности:
 - ◆ Используйте синтаксис NEW / OLD SUBSCRIBE BY.
 - ◆ Обеспечьте эффективный доступ операторов SELECT к базе данных.
- ◆ **Регулярно проверяйте размер журнала транзакций.** Чем больше журнал транзакций, тем дольше Message Agent его сканирует. Регулярно меняйте имя журнала и пользуйтесь параметром DELETE_OLD_LOGS.

Разделение таблиц, не содержащих выражение подписки

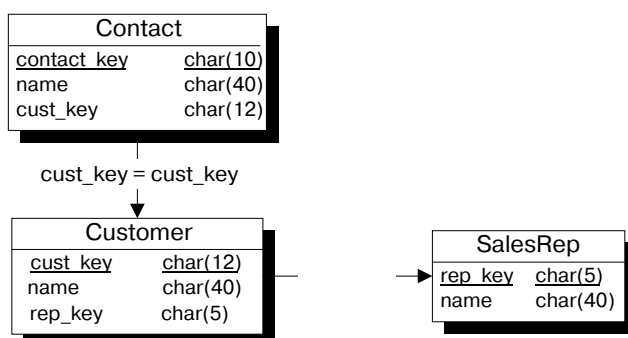
Довольно часто возникает необходимость разделить строки таблицы, даже когда в таблице отсутствует выражение подписки.

Пример базы данных Contact

На примере базы данных Contact иллюстрируются принципы и методы разделения таблиц, не содержащих выражение подписки.

Пример

Ниже в качестве примера представлена простая база данных.



Каждый торговый представитель ведет работу с несколькими клиентам. У некоторых клиентов есть одно контактное лицо, в то время как у других клиентов есть несколько контактных лиц.

Таблицы в базе данных

Ниже приведено более подробное описание этих трех таблиц:

Таблица	Описание
SalesRep	<p>Все торговые представители, работающие в компании. Таблица SalesRep имеет следующие столбцы:</p> <ul style="list-style-type: none"> ♦ rep_key - идентификатор торгового представителя. Этот столбец является первичным ключом. ♦ name - имя торгового представителя. <p>Эта таблица создается с помощью следующего оператора SQL:</p> <pre>CREATE TABLE SalesRep (Rep_key CHAR(12) NOT NULL, Name CHAR(40) NOT NULL, PRIMARY KEY (rep_key))</pre>

Таблица	Описание
Customer	<p>Все клиенты, ведущие дела с данной компанией. Таблица Customer состоит из следующих столбцов:</p> <ul style="list-style-type: none"> ◆ cust_key - идентификатор клиента. Этот столбец является первичным ключом. ◆ name - имя клиента. ◆ rep_key - идентификатор торгового представителя, который работает с данным клиентом. Этот столбец является внешним ключом к таблице SalesRep. <p>Эта таблица создается с помощью следующего оператора SQL:</p> <pre>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))</pre>
Contact	<p>Все отдельные контактные лица, которые имеют дело с компанией. Каждый контакт принадлежит отдельному клиенту. Таблица Contact состоит из следующих столбцов:</p> <ul style="list-style-type: none"> ◆ contact_key - идентификатор контактного лица. Этот столбец является первичным ключом. ◆ name - имя контакта. ◆ cust_key - идентификатор клиента, с которым соотносится данное контактное лицо. Этот столбец является внешним ключом к таблице Customer. <p>Эта таблица создается с помощью следующего оператора SQL:</p> <pre>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))</pre>

Цели репликации

Цель создаваемого проекта репликации данных заключается в обеспечении каждого торгового представителя следующей информацией:

- ◆ Полная таблица **SalesRep**;
- ◆ Информация о закрепленных за торговыми представителями клиентах из таблицы **Customer**;
- ◆ Информация о контактных лицах, закрепленных за соответствующими клиентами, из таблицы **Contact**.

Разделение таблицы Customer в примере базы данных Contact

Таблица **Customer** может быть разделена с использованием значения **rep_key** в качестве выражения подписки. Публикация, которая включает таблицы **SalesRep** и **Customer**, создается следующим образом:

```
CREATE PUBLICATION SalesRepData (
    TABLE SalesRep
```

```
TABLE Customer SUBSCRIBE BY rep_key  
)
```

Разделение таблицы **Contact** в примере базы данных **Contact**

Таблица **Contact** должна быть также разделена между торговыми представителями, однако она не содержит ссылок на значение **rep_key** торговых представителей. Каким образом Message Agent может сопоставить значение подписки со строками этой таблицы, если **rep_key** отсутствует в таблице?

Для решения этой проблемы можно воспользоваться подзапросом в статье **Contact**, который найдет значение для столбца **rep_key** таблицы **Customer**. Публикация в таком случае будет иметь следующий вид:

```
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 )  
)
```

Раздел WHERE в выражении подписки обеспечивает возврат подзапросом только одного значения, так как только одна строка в таблице **Customer** содержит значение **cust_key** из текущей строки таблицы **Contact**.

☞ В консолидированной базе данных Adaptive Server Enterprise используется другое решение данной проблемы. Для получения дополнительной информации см. раздел "Разделение таблиц, не содержащих столбца подписки" на стр. 129.

Перераспределение областей на примере таблицы **Contact**

При **перераспределении областей** происходит переназначение строк между подписчиками. В данном случае перераспределение областей подразумевает под собой переназначение строк в таблице **Customer** и, косвенно, в таблице **Contact**, между торговыми представителями.

Когда клиент переназначается новому торговому представителю, происходит обновление таблицы **Customer**. Оператор UPDATE реплицируется как INSERT или DELETE соответственно для прежнего и нового торгового представителям для того, чтобы строка с информацией о клиенте была правильно перемещена к новому торговому представителю.

☞ Для получения информации о том, каким образом Adaptive Server Anywhere и SQL Remote совместно решают данную проблему, см. раздел "Кто и что получает?" на стр. 73.

При переназначении клиента изменения в таблицу **Contact** не вносятся. Никаких изменений в таблице **Contact** не происходит, поэтому никакие записи относительно таблицы **Contact** в журнал транзакций не добавляются. Из-за отсутствия информации в журнале SQL Remote не может переназначать строки таблиц **Contact** и **Customer**.

Это приводит к возникновению проблем с ссылочной целостностью: таблица **Contact** в удаленной базе данных прежнего торгового представителя содержит значение **cust_key**, для которого **Customer** отсутствует.

Использование триггеров для поддержания таблицы Contacts

Решение заключается в использовании триггера, содержащего специальное выражение оператора UPDATE, который не вносит никаких изменений в таблицы базы данных, но делает запись в журнале транзакций. Эта запись в журнале содержит значения до и после выражения подписки и, таким образом, имеет нужную форму, позволяющую обеспечить правильную репликацию строк в Message Agent.

Триггер должен запускаться до выполнения операций со строкой. Это позволяет вычислить и поместить в журнал значение до репликации (BEFORE). Кроме того, триггер должен запускаться для каждой строки (FOR EACH ROW), а не для каждого оператора, а информация, получаемая триггером, должна быть новым выражением подписки. Message Agent может использовать эту информацию для определения того, какие подписчики получают какие строки.

Определение триггера

Определение триггера следующее:

```
CREATE TRIGGER UpdateCustomer
BEFORE UPDATE ON Customer
REFERENCING NEW AS NewRow
          OLD as OldRow
FOR EACH ROW
BEGIN
    // определение нового выражения подписки
    // для таблицы Customer
    UPDATE Contact
    PUBLICATION SalesRepData
    OLD SUBSCRIBE BY ( OldRow.rep_key )
    NEW SUBSCRIBE BY ( NewRow.rep_key )
    WHERE cust_key = NewRow.cust_key;
END;
```

Специальный оператор UPDATE для публикаций

Оператор UPDATE в этом триггере имеет следующий особый вид:

```
UPDATE имя-таблицы
PUBLICATION имя-публикации
{ SUBSCRIBE BY выражение-подписки }
OLD SUBSCRIBE BY старое-выражение-подписки
NEW SUBSCRIBE BY новое-выражение-подписки }
WHERE условие-поиска
```

Ниже приводится описание значений в разделах оператора UPDATE:

- ◆ *Имя-таблицы* указывает таблицу, которая должна быть изменена в удаленных базах данных.
- ◆ *Имя-публикации* указывает публикацию, для которой должны быть изменены подписки.
- ◆ Значение *выражения-подписки* используется в Message Agent для определения как старого, так и настоящего получателей строк. В качестве альтернативы можно задавать и старое (OLD), и новое (NEW) выражения подписки.
- ◆ Раздел WHERE определяет, какие строки должны быть перемещены между подписанными базами данных.
- ◆ Если триггер использует следующий синтаксис:

```
UPDATE имя-таблицы
PUBLICATION имя-публикации
SUBSCRIBE BY подвыражение
WHERE условие-поиска
```

этот триггер должен быть триггером BEFORE. В данном случае это триггер BEFORE UPDATE. В других контекстах необходимо использовать триггеры BEFORE DELETE и BEFORE INSERT.

- ◆ Если триггер использует альтернативный синтаксис:

Примечания относительно триггера

```
UPDATE имя-таблицы
PUBLICATION имя-публикации
      OLD SUBSCRIBE BY старое-выражение-подписки
      NEW SUBSCRIBE BY новое-выражение-подписки }
WHERE условие-поиска
```

триггер может быть триггером BEFORE или AFTER.

- ◆ Оператор UPDATE заносит в список публикацию и задействованную таблицу. Раздел WHERE в операторе описывает задействованные строки. При использовании этого оператора UPDATE в данные и в саму таблицу никакие изменения не вносятся; происходит только добавление записи в журнале транзакций.
- ◆ Выражение подписки в этом примере возвращает одно значение. Можно также использовать подзапросы, возвращающие множественные значения. Значение выражения подписки должно быть значением после обновления.

В данном случае единственным подписчиком на строку является новый торговый представитель. В разделе "Совместное использование строк в нескольких подписках" на стр. 96 рассмотрены случаи, когда есть и существующие, и новые подписчики.

Информация в журнале транзакций

Ниже описывается информация, которая заносится в журнал транзакций. Ее понимание помогает создавать более эффективные проекты репликации.

- ◆ Допустим, имеются следующие данные:

- ◆ Таблица SalesRep

rep_key	Name
rep1	Ann
rep2	Marc

- ◆ Таблица Customer

cust_key	name	rep_key
cust1	Sybase	rep1
cust2	ASA	rep2

- ◆ Таблица Contact

contact_key	name	cust_key
contact1	David	cust1
contact2	Stefanie	cust2

- ◆ Примените следующий оператор Update для перераспределения областей:

```
UPDATE Customer
SET rep_key = 'rep2'
WHERE cust_key = 'cust1'
```

Исходя из этого оператора, в журнале транзакций появились бы две записи: одна для триггера BEFORE в таблице Contact и одна для фактического UPDATE в таблицу Customer.

```
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 сканирует журнал на наличие данных тэгов. На основе полученной информации он может определить, какие удаленные пользователи получают операторы INSERT, UPDATE или DELETE.

В данном случае списком BEFORE был список **rep1**, а списком AFTER - **rep2**. Если значения списков BEFORE и AFTER отличаются, строки, измененные оператором UPDATE, "перешли" от одного значения подписчика к другому. Это означает, что Message Agent будет посылать оператор DELETE всем удаленным пользователям, которые подписались по значению **rep1** на запись **cust1** таблицы Customer, а также будет посылать оператор INSERT всем удаленным пользователям, которые подписались по значению **rep2**.

Идентичность списков BEFORE и AFTER означает, что удаленный пользователь уже имеет данную строку и что оператор UPDATE будет отправлен.

Совместное использование строк в нескольких подписках

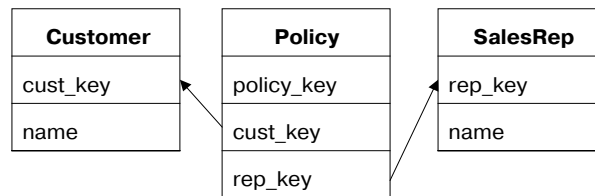
В некоторых случаях может потребоваться включить какую-либо строку в несколько подписок. Например, это происходит при наличии связи типа "многие ко многим". В данном разделе рассматривается конкретный пример, с помощью которого иллюстрируются возможные способы разрешения этой ситуации.

Пример базы данных Policy

Пример с использованием базы данных Policy иллюстрирует, зачем и как разделять таблицы при наличии в базе данных связи "многие ко многим".

Пример базы данных Customer

Здесь в целях иллюстрации приводится простая база данных.



Каждый торговый представитель продает товары нескольким клиентам; некоторые клиенты имеют дело с несколькими торговыми представителями. В данном случае между таблицами **Customer** и **SalesRep** устанавливается связь "многие ко многим".

Таблицы в базе данных

Ниже эти три таблицы рассматриваются более подробно:

Таблица	Описание
SalesRep	<p>Все торговые представители, работающие в компании. Таблица SalesRep имеет следующие столбцы:</p> <ul style="list-style-type: none"> ◆ rep_key - идентификатор торгового представителя. Этот столбец является первичным ключом. ◆ name - имя торгового представителя. <p>Эта таблица создается с помощью следующего оператора SQL:</p> <pre>CREATE TABLE SalesRep (Rep_key CHAR(12) NOT NULL, Name CHAR(40) NOT NULL, PRIMARY KEY (rep_key));</pre>
Customer	<p>Все клиенты, ведущие дела с данной компанией. Таблица Customer состоит из следующих столбцов:</p> <ul style="list-style-type: none"> ◆ cust_key - столбец первичного ключа, содержащий идентификаторы всех клиентов. ◆ name - столбец, содержащий имена всех клиентов. <p>Эта таблица создается с помощью следующего оператора SQL:</p> <pre>CREATE TABLE Customer (Cust_key CHAR(12) NOT NULL, Name CHAR(40) NOT NULL, PRIMARY KEY (cust_key));</pre>

Таблица	Описание
Policy	<p>Таблица из трех столбцов, обеспечивающая связь "многие ко многим" между клиентами и торговыми представителями. Таблица Policy состоит из следующих столбцов:</p> <ul style="list-style-type: none"> ◆ policy_key - столбец первичного ключа, содержащий идентификаторы для связи между клиентом и торговым представителем. ◆ cust_key - столбец, содержащий идентификатор контактного лица клиента в связи между клиентом и торговым представителем. ◆ rep_key - столбец, содержащий идентификатор торгового представителя в связи между клиентом и торговым представителем. <p>Эта таблица создается с помощью следующего оператора SQL:</p> <pre>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));</pre>

Цели репликации

Цель производимой репликации данных заключается в обеспечении каждого торгового представителя следующей информацией:

- ◆ Полная таблица **SalesRep**;
- ◆ Строки из таблицы **Policy**, включающие в себя связи между клиентами и торговыми представителями, в которых задействован торговый представитель, подписавшийся на эти данные;
- ◆ Строки из таблицы **Customer**, в которых перечисляются клиенты, имеющие деловые контакты с торговым представителем, подписавшимся на эти данные.

Новые проблемы

Наличие связи "многие ко многим" между клиентами и торговыми представителями приводит к появлению новых проблем, связанных с необходимостью обеспечения совместного использования информации:

- ◆ Одна из таблиц (в данном случае это таблица **Customer**) не имеет ссылок на значение торгового представителя, которое используется в подписках для разделения данных.

Как и прежде, для решения проблемы используется подзапрос в публикации.

- ◆ Каждая строка в таблице **Customer** может быть связана со многими строками в таблице **SalesRep** и совместно использоваться многими базами данных торговых представителей.

Другими словами, строки таблицы **Contact**, приведенной на стр. 103 в разделе "Разделение таблиц, не содержащих выражение подписки", были разделены публикацией на неперекрывающиеся наборы. В настоящем примере имеют место перекрывающиеся подписки.

Для достижения цели репликации снова потребуется одна публикация и набор подписок. В данном примере используются два триггера для обработки процедуры перевода клиентов от одного торгового представителя к другому.

Публикация

Отдельно взятая публикация создает основу для совместного использования данных:

```
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  
    ),  
);
```

Операторы подписки в данном примере точно такие же, как и в предыдущем примере.

Принципы работы публикации

Публикация включает в себя часть или всю информацию из каждой из приведенных трех таблиц. Для понимания принципов работы публикации необходимо последовательно проанализировать каждую статью.

- ◆ **Таблица SalesRep.** Спецификаторы к данной статье отсутствуют, поэтому в публикацию включена вся таблица **SalesRep** целиком.

```
...  
    TABLE SalesRep,  
...
```

- ◆ **Таблица Policy.** В данной статье используется выражение подписки для определения столбца, используемого для разделения данных между торговыми представителями:

```
...  
    TABLE Policy  
    SUBSCRIBE BY rep_key,  
...
```

Выражение подписки обеспечивает получение каждым торговым представителем только тех строк таблицы, для которых значение столбца **rep_key** соответствует значению, указанному в подписке.

Разделение таблицы **Policy** является **разделением без перекрытия**: не существует строк, совместно используемых более чем одним подписчиком.

- ◆ **Таблица Customer.** Для определения разделения используется выражение подписки с подзапросом. Параметры статьи задаются следующим образом:

```
...  
    TABLE Customer SUBSCRIBE BY (  
        SELECT rep_key  
        FROM Policy  
        WHERE Policy.cust_key =  
            Customer.cust_key  
    ),  
...
```

Разделение таблицы **Customer** является **разделением с перекрытием**: некоторые строки совместно используются более чем одним подписчиком.

Многозначные подзапросы в публикациях

Подзапрос в статье **Customer** возвращает в своем результирующем наборе единственный столбец (**rep_key**), однако может возвращать множество строк, соответствующих всем тем торговым представителям, которые работают с конкретным клиентом. Когда выражение подписки имеет множество значений, строка реплицируется всем подписчикам, в подписках которых указано любое из этих значений. Именно эта возможность использования многозначных выражений подписки позволяет выполнять разделение таблицы с перекрытием.

Перераспределение областей со связью "многие ко многим"

Проблема перераспределения областей (переназначение строк между подписчиками) требует особого рассмотрения - так же, как и в разделе "Перераспределение областей на примере таблицы Contact" на стр. 92.

При разрешенном перераспределении областей (переназначении строк между подписчиками) необходимо написать триггеры для сохранения целостности данных во всей системе репликации.

Перемещение информации о клиентах

В данном примере требуется переместить информацию о клиенте путем удаления и вставки строк в таблице **Policy**.

Чтобы аннулировать связь между клиентом и торговым представителем, необходимо удалить строку в таблице **Policy**. В этом случае изменение в таблице **Policy** реплицируется торговому представителю правильно, и строка удаляется из соответствующей базы данных. Однако в таблице **Customer** изменений не происходит, и, соответственно, таблица **Customer** реплицируется подписчику без каких-либо изменений.

При отсутствии триггеров это привело бы к наличию у подписчика неверных данных в таблице **Customer**. Такая же проблема возникает при добавлении в таблицу **Policy** новой строки.

Использование триггеров для решения проблемы

Решение проблемы заключается в написании триггеров, которые запускаются после внесения изменений в таблицу **Policy**, с использованием специального синтаксиса оператора UPDATE. Специальный оператор UPDATE не вносит какие-либо изменения в таблицы базы данных, но добавляет определенную запись в журнал транзакций, которую SQL Remote использует для поддержания согласованности данных в базах данных подписчиков.

Триггер BEFORE INSERT

Ниже приводится триггер, который отслеживает изменения, сделанные оператором INSERT в таблице **Policy**, обеспечивая тем самым согласованность данных в удаленных базах данных.

```
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;
```

Триггер BEFORE DELETE

Следующий триггер отслеживает изменения, производимые оператором DELETE в таблице **Policy**:

```
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;
```

Некоторые особенности этого триггера схожи с теми, что были описаны в предыдущем разделе. Главной новой функцией является то, что триггер INSERT содержит подзапрос, и этот подзапрос может иметь множество значений.

Многозначные подзапросы

Подзапросом (возможно, многозначным) в триггере BEFORE INSERT является выражение UNION:

```
...  
SELECT rep_key  
FROM Policy  
WHERE cust_key = NewRow.cust_key  
UNION ALL  
SELECT NewRow.rep_key  
...
```

- ♦ Второй частью выражения UNION является значение **rep_key** нового торгового представителя для данного клиента, информация о котором берется из оператора INSERT.
- ♦ В первую часть выражения UNION входит совокупная информация о существующих торговых представителях, работающих с данным клиентом, информация о котором в свою очередь берется из таблицы Policy.

Данный пример иллюстрирует тот факт, что результирующий набор запроса подписки должен содержать информацию обо всех торговых представителях, получающих данную строку, а не только информацию о новых торговых представителях.

Подзапрос в триггере BEFORE DELETE является многозначным:

```
...  
SELECT rep_key  
FROM Policy  
WHERE cust_key = OldRow.cust_key  
AND rep_key <> OldRow.rep_key  
...
```

- ♦ Подзапрос берет значения **rep_key** из таблицы **Policy**. К этим значениям относятся значения первичного ключа всех торговых представителей, работающих с клиентом, информация о котором подлежит перемещению (**WHERE cust_key = OldRow.cust_key**), за исключением торгового представителя, информация о котором удаляется (**AND rep_key <> OldRow.rep_key**).

Данным примером еще раз подчеркивается тот факт, что результирующий набор запроса подписки должен содержать все совпадающие значения торговых представителей, получающих данную строку, после выполнения триггера DELETE.

Примечания

- ♦ Данные в таблице **Customer** не отождествляются с отдельным подписчиком (по значению первичного ключа, например) и совместно используются несколькими подписчиками. Это дает возможность обновлять данные на нескольких удаленных узлах между сеансами передачи сообщений репликации. Однако такое обновление может привести к возникновению конфликтов. Существует два подхода к решению данной проблемы: либо путем применения системы полномочий (к примеру, наделением определенных пользователей правом обновлять таблицу Customer), либо добавлением к базе данных триггеры RESOLVE UPDATE для решения конфликтов программным путем.
- ♦ В данном разделе не рассматривается работа триггеров UPDATE с таблицей Policy. Необходимо либо предотвращать их появление, либо использовать триггер BEFORE UPDATE, который сочетает в себе функции триггеров BEFORE INSERT и BEFORE DELETE, рассмотренных в примере.

Использование параметра `Subscribe_by_remote` со связями "многие ко многим"

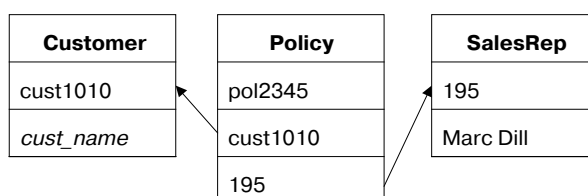
Когда параметр `Subscribe_by_remote` установлен в значение `ON`, при выполнении операций с удаленных баз данных над строками с нулевым значением параметра `SUBSCRIBE BY` или же с пустой строкой считается, что удаленный пользователь подписан на данную строку. По умолчанию параметр `Subscribe_by_remote` установлен в `ON`. В большинстве случаев рекомендуется использовать именно это значение.

С помощью параметра `Subscribe_by_remote` решается проблема, которая в противном случае возникла бы с некоторыми публикациями, в том числе и с таблицей `Policy` из предыдущего примера. В данном разделе приводится описание данной проблемы и способа автоматического предотвращения ее возникновения с помощью данного параметра.

В публикации используется подзапрос к выражению подписки таблицы **Customer**, поскольку каждый клиент (из таблицы `Customer`) может быть связан с несколькими торговыми представителями (из таблицы `SalesReps`).

```
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
    ),
);
```

Марк Дилл (Marc Dill) - торговый представитель, который только что начал работать с новым клиентом. Он вставляет новую строку в таблицу **Customer** и еще одну строку в таблицу **Policy**, чтобы закрепить за собой нового клиента.



Поскольку в консолидированной базе данных операцию `INSERT` со строкой в таблице `Customer` выполняет `Message Agent`, при выполнении операции `INSERT` `Adaptive Server Anywhere` заносит значение подписки в журнал транзакций.

Позже, когда `Message Agent` сканирует журнал, он формирует список подписчиков из выражения подписки, но Марка Дилла нет в этом списке, так как строка в таблице `Policy`, посредством которой клиент закрепляется за этим торговым представителем, еще не была обработана. Если бы параметр `Subscribe_by_remote` был установлен в значение `OFF`, информация о новом клиенте отсылалась бы назад к Марку Диллу как результат выполнения операции `DELETE`.

Если параметр `Subscribe_by_remote` установлен в значение `ON`, `Message Agent` считает, что строка принадлежит тому торговому представителю, который внес ее в таблицу; `INSERT` не реплицируется обратно к Марку Диллу, и данные в системе репликации сохраняют согласованность.

При установке параметра `Subscribe_by_remote` в значение `OFF` необходимо обеспечить то, что строка `Policy` вставляется до строки `Customer`; при этом ссылочная целостность обеспечивается установкой проверяющих условий в конец транзакции.

Разрешение конфликтов

Конфликт при выполнении операций обновления (UPDATE) происходит при возникновении следующей последовательности событий:

- 1 Пользователь 1 обновляет строку на удаленном узле 1.
- 2 Пользователь 2 обновляет ту же самую строку на удаленном узле 2.
- 3 Обновление от пользователя 1 реплицируется в консолидированную базу данных.
- 4 Обновление от пользователя 2 реплицируется в консолидированную базу данных.

При репликации операторов UPDATE в SQL Remote Message Agent каждый оператор UPDATE выполняется отдельно для каждой строки. Кроме того, сообщение также содержит старые значения строки для сравнения. Когда консолидированная база данных получает обновление от пользователя 2, значения в строке не совпадают с теми, что записаны в сообщении.



Разрешение конфликтов по умолчанию

По умолчанию выполнение UPDATE все же продолжается, с тем, чтобы обновление от пользователя 2 (последнее обновление, поступившее в консолидированную базу данных) было внесено в консолидированную базу данных и реплицировано во все остальные базы данных пользователей, подписанных на данную строку.

В общем случае, применяемый по умолчанию метод решения конфликта заключается в выполнении более поздней операции (в нашем случае это операция обновления от пользователя 2), при этом конфликт не регистрируется. Обновление от пользователя 1 не сохраняется. SQL Remote также позволяет применять другие решения конфликта с использованием для этих целей триггера, который дает возможность изменять данные нужным образом.

Разрешение конфликтов не применимо к обновлениям первичного ключа
 Конфликты операторов UPDATE не возникают в связи с обновлением первичного ключа, поскольку в инсталляции SQL Remote обновление первичных ключей производить не следует. Возможность возникновения конфликтов первичного ключа должна быть исключена на этапе настройки системы.

В данном разделе описываются методы разрешения конфликтов в консолидированной базе данных в инсталляции SQL Remote.

Принципы разрешения конфликтов в SQL Remote

При обнаружении конфликта

Сообщения репликации SQL Remote включают в себя операторы UPDATE в виде набора обнаруженных обновлений отдельной строки, где каждое обновление имеет раздел VERIFY, который содержит значения до обновления.

Конфликт обновлений фиксируется сервером базы данных в случае, когда не удается сопоставить значения раздела VERIFY со строками в базе данных.

Обнаружение и разрешение конфликтов выполняет Message Agent, но только в консолидированной базе данных. При обнаружении конфликта операторов UPDATE в сообщении от удаленной базы данных Message Agent активизирует на сервере базы данных две операции:

- 1 Запускаются любые триггеры разрешения конфликтов (RESOLVE UPDATE).
- 2 Выполняется оператор UPDATE.

Операторы UPDATE выполняются даже в том случае, если значения раздела VERIFY не совпадают, независимо от наличия или отсутствия триггера UPDATE RESOLVE.

Процесс разрешения конфликтов может проходить по-разному. Например,

- ◆ В некоторых приложениях процедура разрешения конфликта заключается в пересылке сообщения о конфликте в таблицу.
- ◆ Может потребоваться установить приоритетность обновлений, выполняемых в консолидированной базе данных, над обновлениями, производимыми на удаленных узлах.
- ◆ Процесс разрешения конфликтов может быть более сложным, как, например, при распределении инвентарных номеров при поставках и заказе товаров.

☞ В консолидированной базе данных Adaptive Server Enterprise применяется другой метод разрешения конфликтов. Для получения дополнительной информации см. раздел "Принципы разрешения конфликтов в SQL Remote" на стр. 103.

Реализация методов разрешения конфликтов

В данном разделе описываются возможности по реализации пользовательских методов разрешения конфликтов в SQL Remote для Adaptive Server Anywhere. Основные принципы решения этих конфликтов такие же, как и в SQL Remote для Adaptive Server Enterprise, однако их реализация осуществляется по-разному.

Для управления разрешением конфликтов UPDATE в SQL Remote предусмотрена возможность определения **триггеров разрешения конфликтов**. Триггеры разрешения конфликтов запускаются только в консолидированной базе данных при поступлении в базу сообщений от удаленных пользователей. При обнаружении конфликта UPDATE в консолидированной базе данных выполняется следующая последовательность действий.

- 1 Запускаются любые триггеры разрешения конфликтов, определенные для данной операции.
- 2 Выполняется оператор UPDATE.

- 3 Любые действия триггера, а так же оператора UPDATE, реплицируются во все удаленные базы данных, включая БД отправителя сообщения, вызвавшего возникновение конфликта.

Вообще, SQL Remote для Adaptive Server Anywhere не реплицирует действия триггеров: предполагается, что в удаленной базе данных необходимый триггер уже имеется. Триггеры разрешения конфликтов запускаются только в консолидированных базах данных, поэтому их действия реплицируются в удаленные базы данных.

- 4 Триггеры RESOLVE UPDATE не запускаются в удаленных базах данных, если сообщение от консолидированной базы данных содержит конфликт UPDATE.
- 5 Операция UPDATE выполняется в удаленных базах данных.

По завершении данного процесса в ходе настройки данные унифицируются.

Конфликты UPDATE не могут возникать там, где данные совместно используются для чтения, но каждая строка (поскольку она идентифицируется по своему первичному ключу) обновляется только на одном узле. Подобные конфликты происходят тогда, когда данные одновременно обновляются на нескольких узлах.

Использование триггеров разрешения конфликтов

В данном разделе описывается применение триггера RESOLVE UPDATE, или триггеров **разрешения конфликтов**.

Операторы UPDATE с разделом VERIFY

Триггеры разрешения конфликтов запускаются в случае, если перед обновлением значения в разделе VERIFY оператора UPDATE и значения в базе данных не совпадают. Оператор UPDATE с разделом VERIFY принимает следующий вид:

```
UPDATE список-таблиц
SET имя-столбца = выражение, ...
[ FROM список-таблиц ]
[ VERIFY (имя-столбца, ...)
  VALUES ( выражение, ...) ]
[ WHERE условие-поиска ]
```

Раздел VERIFY сравнивает значения определенных столбцов с набором ожидаемых значений, которые присутствовали в базе данных издателя на момент выполнения там оператора UPDATE.

Раздел VERIFY используется только для обновлений отдельных строк. Однако Message Agent реплицирует введенные в базу данных многострочные операторы обновления в виде набора обновлений отдельных строк, поэтому это не налагает никаких ограничений на клиентские приложения.

Синтаксис триггера разрешения конфликтов

Синтаксис триггера RESOLVE UPDATE следующий:

```
CREATE TRIGGER имя-триггера
RESOLVE UPDATE
OF имя-столбца ON имя-таблицы
[ REFERENCING [ OLD AS старое_значение ]
  [ NEW AS новое_значение ]
  [ REMOTE AS значение_удаленного_узла ] ]
FOR EACH ROW
BEGIN
...
END
```

Триггеры RESOLVE UPDATE запускаются перед обновлением каждой строки. Раздел REFERENCING предоставляет доступ к значениям в строке таблицы, которую необходимо обновить (OLD), к значениям, на которые строка должна быть изменена (NEW), и к строкам, которые должны присутствовать согласно

Использование параметра VERIFY_ALL_COLUMNS MNS

разделу VERIFY (REMOTE). В разделе REMOTE AS ссылаться можно только на столбцы, которые присутствуют в разделе VERIFY; при обращении к другим столбцам возникает ошибка “column not found” (“столбец не найден”).

По умолчанию для параметра базы данных VERIFY_ALL_COLUMNS установлено значение OFF. Если установить для него значение ON, все столбцы будут проверяться на наличие реплицированных обновлений, и триггер RESOLVE UPDATE будет запускаться каждый раз, когда обнаружится несоответствие в каком-либо из столбцов. Если для этого параметра установить значение OFF, проверяться будут только те столбцы, которые обновляются.

Установка данного параметра в значение ON приведет к увеличению размера сообщений, так как для каждого обновления будет пересылаться больше информации.

Если данный параметр устанавливается в консолидированной базе данных перед извлечением удаленных баз данных, в удаленных базах данных он так же будет установлен.

Установить параметр VERIFY_ALL_COLUMNS можно как для группы PUBLIC, так и только для одного пользователя, информация о котором содержится в строке подключения Message Agent.

Использование специальной константы CURRENT_REMOTE_USER

Специальная константа CURRENT_REMOTE_USER хранит идентификатор удаленного пользователя, посылающего сообщение. Она может быть использована в триггерах RESOLVE UPDATE, которые помещают в таблицу сообщения о конфликтах, для идентификации пользователя, инициировавшего конфликт.

Примеры разрешения конфликтов

В данном разделе описываются несколько способов использования триггеров RESOLVE UPDATE для разрешения конфликтов.

Разрешение конфликтов дат

Предположим, что в таблице в системе управления контактами имеется столбец с информацией о самых последних контактах с каждым клиентом.

Один представитель проводит переговоры с клиентом в пятницу, но не загружает необходимые изменения в консолидированную базу данных до следующего понедельника. Между тем, второй представитель встречается с клиентом в субботу, после чего вечером того же дня вносит соответствующие изменения в базу данных.

При репликации в консолидированную базу данных обновления, сделанного в субботу, конфликтов не происходит, но когда база данных получает следующее обновление в понедельник, обнаруживается, что соответствующая строка уже изменена.

По умолчанию, обновление, сделанное в понедельник, будет обработано, и в столбец будет внесена неверная информация о том, что последний контакт имел место в пятницу.

Конфликты обновления с данным столбцом необходимо решать путем вставки наиболее поздней даты в соответствующую строку.

Реализация решения

Следующий триггер RESOLVE UPDATE выбирает наиболее позднее из двух новых значений и вводит его в базу данных.

```
CREATE TRIGGER contact_date RESOLVE UPDATE
ON contact
REFERENCING OLD AS old_name
NEW AS new_name
```

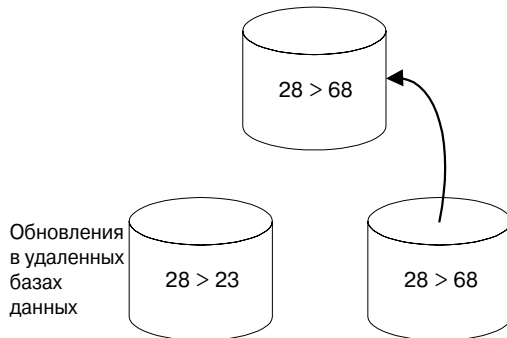
```
FOR EACH ROW
BEGIN
  IF new_name.contact_date <
    old_name.contact_date THEN
    SET new_name.contact_date
      = old_name.contact_date
  END IF
END
```

Если обновляемое значение является более поздним по дате, нежели значение в обновлении, новое значение удаляется, и изменения в запись не вносятся.

Разрешение конфликтов инвентаризации

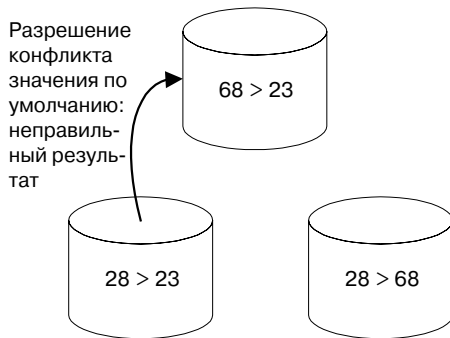
Рассмотрим складскую систему изготовителя спортивных товаров. Существует таблица с информацией о товарах, в которой столбец **quantity** содержит количество каждого товара на складе. Обновление этого столбца соответственно либо уменьшит количество на складе, либо, если поступила новая партия товара, добавит его.

Торговый представитель вводит заказ в удаленной базе данных, уменьшая запас рубашек на 5, с 28 до 23, и вводит эту информацию в своей базе данных. Тем временем, перед тем, как это обновление было реплицировано в консолидированную базу данных, поступает новая партия рубашек, и склад принимает эту партию, добавляя 40 в столбец **quantity**, что в сумме дает 68.

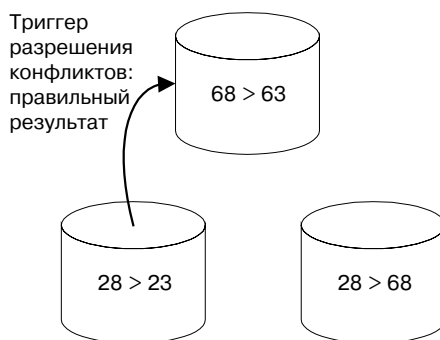


Складская запись вводится в базу данных: столбец **quantity** теперь показывает, что на складе есть 68 рубашек. Когда в базу данных поступает обновление от торгового представителя, оно вызывает конфликт - Adaptive Server Anywhere обнаруживает, что обновление происходит с 28 на 23, но текущее значение столбца равно 68.

По умолчанию, самое последнее обновление выполняется, и количество товара устанавливается на неправильное значение 23.



В данной ситуации конфликт должен быть решен путем суммирования изменений в столбце запасов, чтобы вывести конечный результат и установить в базе данных окончательное значение 63.



Реализация решения

Подходящий для данной ситуации триггер RESOLVE UPDATE добавляет приращения из двух обновлений. Например:

```
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
```

До выполнения оператора UPDATE этот триггер добавляет различие между старым значением в консолидированной базе данных (68) и старым значением в удаленной базе данных на момент, когда выполнялся первоначальный оператор UPDATE (28), к новому присланному значению. Таким образом, **new_val.quantity** становится равен 63 (= 23 + 68-28); это значение и вводится в столбец **quantity**.

Согласованность в удаленной базе данных обеспечивается следующим образом:

- 1 Первоначальный удаленный оператор UPDATE изменил значение с 28 на 23.
- 2 Запись со склада реплицируется в удаленную базу данных, но не обрабатывается, поскольку старое значение не соответствует ожидаемому значению.
- 3 Изменения, сделанные триггером RESOLVE UPDATE, реплицируются в удаленную базу данных.

Сообщения о конфликтах

В некоторых случаях можно не вносить изменений в метод разрешения конфликтов, принятый в SQL Remote по умолчанию; возможно, будет просто необходимо составлять сообщения о конфликтах и сохранять их в таблице.

В этом случае достаточно просмотреть таблицу конфликтов, чтобы выяснить, были ли конфликты и какие, и, если необходимо, предпринять соответствующие действия для их разрешения.

Методы предотвращения ошибок ссылочной целостности

Таблицы в реляционной базе данных связываются посредством ссылок по внешнему ключу. Ограничения ссылочной целостности, применяемые как следствие использования этих ссылок, обеспечивают постоянную непротиворечивость информации в базе данных. При репликации только части базы данных могут возникнуть проблемы сохранения ссылочной целостности реплицированной базы данных.

Ошибки при
отсутствии
репликации
связанной таблицей

Этих проблем можно избежать, если уделять должное внимание вопросам ссылочной целостности еще на этапе построения публикации. В данном разделе описываются некоторые наиболее общие проблемы ссылочной целостности и рассматриваются способы их предотвращения.

Публикация **sales**, описанная в разделе "Публикация целых таблиц" на стр. 79, включает в себя таблицу **sales_order**:

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

Таблица **sales_order** содержит внешний ключ к таблице **employee**. Идентификатор торгового представителя является внешним ключом в таблице **sales_order**, ссылающимся на первичный ключ таблицы **employee**. Однако таблица **employee** не включена в публикацию.

Если публикация создается подобным способом, данные новых заказов на закупку не реплицируются до тех пор, пока в удаленной базе данных не удаляются ссылки по внешнему ключу из таблицы **sales_order**.

При использовании утилиты извлечения для создания удаленных баз данных ссылка по внешнему ключу автоматически исключается из удаленной базы данных, и подобной проблемы не возникает. Однако в базе данных не существует ограничений, которые предотвращали бы вставку недопустимых значений в столбец **rep_id** таблицы **SalesRep**; если такое случается, оператор **INSERT** в консолидированной базе данных выполняться не будет. Для предотвращения возникновения подобной проблемы можно включить в публикацию таблицу **Employee** (или, по крайней мере, ее первичный ключ).

Создание триггеров для предотвращения ошибок

Действия, выполняемые триггерами, не реплицируются: при выполнении репликации предполагается, что триггеры, существующие в одной базе данных системы SQL Remote, существуют и в других базах данных этой системы. Когда действие, запускающее триггер в консолидированной базе данных, реплицируется в узел репликации, триггер запускается автоматически. По умолчанию утилита извлечения базы данных извлекает определение триггера с тем, чтобы они также присутствовали и в удаленной базе данных.

Если публикация включает в себя только поднабор базы данных, триггер в консолидированной базе данных может ссылаться на таблицы или строки, которые есть в консолидированной базе данных, но которые отсутствуют в удаленных базах данных. Можно построить триггер так, чтобы он не допускал подобных ошибок, сделав его действия условными с помощью оператора **IF**. В приведенном ниже списке предлагается несколько способов создания триггера для работы с консолидированной и удаленными базами данных.

- ◆ Установка условий для выполнения действий триггера со значением **CURRENT PUBLISHER**. В этом случае триггер не будет выполнять некоторые действия в удаленной базе данных.
- ◆ Установка условий для выполнения действий с функцией **object_id**, не возвращающей **NULL**. Функция **object_id** принимает таблицу или другой объект как аргумент и возвращает идентификатор этого объекта или **NULL**, если объект не существует.
- ◆ Установка условий для выполнения действий над оператором **SELECT**, который определяет, существуют ли строки.

Триггер RESOLVE UPDATE является специальным триггером, предназначенным для разрешения конфликтов операторов UPDATE; он рассматривается в разделе "Примеры разрешения конфликтов" на стр. 105. Действия триггеров RESOLVE UPDATE реплицируются в удаленные базы данных, включая базу данных, вызвавшую возникновение конфликта.

Обеспечение уникальности первичных ключей

Значения первичного ключа должны быть уникальны. Когда все пользователи подключаются к одной и той же базе данных, проблем с поддержанием уникальности значений не существует. При попытке пользователя повторно использовать то же значение оператор INSERT не выполняется.

По-другому обстоит ситуация с системой репликации, поскольку пользователи подключаются ко многим базам данных. Проблема может возникнуть тогда, когда два пользователя, подключенные к разным базам данных, вставляют строку, используя при этом одно и то же значение первичного ключа. Оба оператора этих пользователей выполняются, так как в каждой базе данных это значение уникально.

Проблемы в системе репликации возникают, однако, тогда, когда два пользователя, подключенные к различным базам данных, вставляют строку с помощью оператором INSERT, используя одно и то же значение первичного ключа. Оператор INSERT, который поступает в данную базу данных в системе репликации последним из двух, не выполняется. Поскольку SQL Remote является системой репликации для пользователей, выполняющих подключение к базе данных периодически, механизм блокирования для всех баз данных системы репликации может не действовать. Необходимо построить систему SQL Remote таким образом, чтобы ошибки дублирования первичных ключей не возникали.

Для предотвращения ошибок первичных ключей в системе SQL Remote, необходимо обеспечить уникальность первичных ключей таблиц, которые могут быть изменены с нескольких узлов. Для достижения данной цели существует несколько способов. В этой главе рассматриваются два обобщенных, целесообразных и надежных способа.

- 1 С помощью используемой по умолчанию функции автоприращения глобальной переменной Adaptive Server Anywhere;
- 2 С использованием пулов первичных ключей для поддержания на каждом узле списка неиспользуемых значений первичных ключей.

Для предотвращения дублирования значений можно использовать эти способы отдельно или совместно.

Использование значений по умолчанию в столбце глобального автоприращения

В Adaptive Server Anywhere можно установить значение GLOBAL AUTOINCREMENT как значение, используемое в столбце по умолчанию. Можно применять это значение по умолчанию для любого столбца, в котором требуется поддерживать уникальность значений, однако его особенно полезно использовать с первичными ключами. Эта функция позволяет упростить задачу генерации уникальных значений в системах репликации данных между различными БД, как правило, посредством синхронизации MobiLink.

При установке глобального автоприращения по умолчанию область значений для данного столбца разделяется. Каждый раздел содержит равное количество значений. Например, если размер раздела для целочисленного столбца в базе данных устанавливается в 1000, один раздел будет с 1001 по 2000, следующий с 2001 по 3000 и т. д.

Для каждой копии базы данных задается уникальный глобальный идентификационный номер базы данных. Adaptive Server Anywhere предоставляет значения по умолчанию в базе данных только из раздела, однозначно идентифицированного номером этой базы данных.

Например, если базе данных в приведенном выше примере назначается идентификационный номер 10, значения по умолчанию в этой базе данных выбирались бы в диапазоне 10001-11000. Для другой копии базы данных, которой назначен идентификационный номер 11, значение по умолчанию для этого же столбца будет находиться в диапазоне 11001-12000.

Объявление глобального автоприращения по умолчанию

В базе данных значения по умолчанию можно устанавливать путем выбора свойств столбца в Sybase Central или включения фразы DEFAULT GLOBAL AUTOINCREMENT в оператор TABLE или ALTER TABLE.

Дополнительно можно указывать размер раздела в скобках сразу же после ключевого слова AUTOINCREMENT. Размер раздела может быть задан любым положительным целым числом, хотя, как правило, размер раздела выбирается так, чтобы запас чисел в пределах одного раздела редко или совсем не истощался.

Для столбцов типа INT или UNSIGNED INT размер раздела по умолчанию равен $2^{16} = 65536$; для столбцов других типов размер раздела по умолчанию составляет $2^{32} = 4294967296$. Поскольку эти значения по умолчанию могут оказаться неподходящими, особенно если столбец имеет другой тип, нежели INT или BIGINT, рекомендуется указывать размер раздела явно.

Например, нижеприведенный оператор создает простую таблицу с двумя столбцами: для целых чисел, в которых хранятся идентификационные номера клиентов, и для символьных строк, в которых содержатся имена клиентов.

```
CREATE TABLE customer (
    id INT DEFAULT GLOBAL AUTOINCREMENT (5000)
    name VARCHAR(128) NOT NULL
    PRIMARY KEY (id)
)
```

В приведенном выше примере выбран размер раздела 5000.

☞ Для получения дополнительной информации о GLOBAL AUTOINCREMENT см. раздел "Оператор CREATE TABLE" (CREATE TABLE statement) на стр. 350 в документе *"Справочное руководство по SQL для ASA" (ASA SQL Reference Manual)*.

Установка значения Global_database_id

При развертывании приложения необходимо назначать различные идентификационные номера каждой базе данных. Для создания и распределения идентификационных номеров можно использовать ряд способов. Один способ заключается в том, что значения вносятся в таблицу, и необходимая строка загружается в каждую базу данных на основе какого-то другого уникального свойства БД, например, имени пользователя.

❖ Процедура установки глобального идентификационного номера базы данных

- ◆ Идентификационный номер базы данных назначается путем установки значения общедоступного параметра **Global_database_id**. Идентификационный номер должен быть неотрицательным целым числом.

Например, приведенный ниже оператор устанавливает идентификационный номер базы данных 20.

```
SET OPTION PUBLIC.Global_database_id = 20
```

Если размер раздела для определенного столбца равен 5000, значение по умолчанию для этой базы данных выбирается из диапазона 100001–105000.

Установка уникальных идентификационных номеров баз данных при извлечении баз данных

Если для создания удаленных баз данных используется утилита извлечения, в целях автоматизации задачи можно написать хранимую процедуру. Если имеется хранимая процедура под названием `sp_hook_dbxtract_begin`, она автоматически вызывается утилитой извлечения. Перед вызовом процедуры утилита извлечения создает временную таблицу под названием `#hook_dict`, в которой содержится следующее:

name	value
extracted_db_global_id	извлекаемый идентификатор пользователя

Если процедура `sp_hook_dbxtract_begin` создается для обновления столбца `value` строки, это значение используется в качестве параметра `GLOBAL_DATABASE_ID` извлеченной базы данных, при этом оно отмечает начало диапазона значений первичного ключа для значений `GLOBAL DEFAULT AUTOINCREMENT`.

Пример

Рассмотрим пример извлечения базы данных для удаленного пользователя `user2` с `user_id` равным 101.

Если не определять процедуру `sp_hook_dbxtract_begin`, для параметра `Global_database_id` извлеченной базы данных будет установлено значение 101.

При определении процедуры `sp_hook_dbxtract_begin` таким образом, чтобы она не вносила изменения в строки таблицы `#hook_dict`, для параметра все равно будет устанавливаться значение 101.

При следующей настройке базы данных:

```
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'
```

каждая извлеченная или повторно извлеченная база данных получит разный `Global_database_id`. Первый начинается с 1001, следующий с 2001 и т. д.

Для упрощения отладки процедур, подключаемых через адрес, при работе `dbxtract` в режиме расширенного вывода может быть получено следующее:

- ◆ найденные процедуры, подключенные через адрес;
- ◆ содержимое таблицы `#hook_dict` перед вызовом процедуры, подключенной через адрес;
- ◆ содержимое таблицы `#hook_dict` после вызова процедуры, подключенной через адрес.

Выбор значений по умолчанию

Общедоступный параметр `Global_database_id` в каждой базе данных должен быть установлен на уникальное неотрицательное целое число. Диапазон значений по умолчанию для отдельной базы данных находится в пределах от $pn + 1$ до $p(n + 1)$, где p - размер раздела, и n - значение общедоступного параметра `Global_database_id`. Например, если размер раздела равен 1000, и `Global_database_id` установлен в значение 3, то диапазон будет от 3001 до 4000.

Если параметр **Global_database_id** установлен на неотрицательное целое число, Adaptive Server Anywhere выбирает значения по умолчанию, применяя следующие правила:

- ◆ Если столбец не содержит значений в текущем разделе, первым значением по умолчанию является $pn + 1$.
- ◆ Если столбец содержит значения в текущем разделе, но все они меньше чем $p (n + 1)$, следующим значением по умолчанию будет значение на один большее, чем предыдущее максимальное значение в этом диапазоне.
- ◆ Значения столбца по умолчанию не зависят от значения в столбце за пределами текущего раздела; т. е. от чисел меньше чем $pn + 1$ или больше чем $p (n + 1)$. Такие значения могут присутствовать, если они были реплицированы из другой базы данных посредством синхронизации MobiLink.

Если для общедоступного параметра **Global_database_id** установлено значение по умолчанию 2147483647, то в столбец вставляется значение NULL. Если использование значений NULL не допускается, попытка вставки строки приводит к ошибке. Подобная ситуация возникает, например, если столбец указан в первичном ключе таблицы.

Поскольку для общедоступного параметра **Global_database_id** не могут быть установлены отрицательные значения, выбираемые значения всегда положительны. Максимальный идентификационный номер ограничивается только типом данных столбца и размером раздела.

Значения по умолчанию NULL генерируются также тогда, когда полностью исчерпывается запас значений внутри раздела. В этом случае базе данных должно быть присвоено новое значение параметра **Global_database_id**, с тем, чтобы значения по умолчанию выбирались из другого раздела. Если столбец не допускает использования нулевых значений, попытка вставки значения NULL приводит к ошибке. Для определения низкого уровня запаса неиспользованных значений и принятия соответствующих мер следует создать событие типа **GlobalAutoincrement**.

Если запас значений в отдельном разделе исчерпывается, для этой базы данных можно назначать новый идентификатор базы данных. Назначать новые идентификационные номера баз данных можно любым подходящим способом. Один из возможных способов заключается в поддержании пула неиспользованных значений идентификаторов баз данных. Этот пул поддерживается таким же способом, что и пул первичных ключей.

Можно задать обработчик событий, чтобы автоматически уведомлять администратора базы данных (или чтобы выполнять какое-либо другое действие) при обнаружении низкого уровня запаса значений в разделе. Для получения дополнительной информации см. раздел "Определение условий триггеров для событий" (Defining trigger conditions for events) на стр. 237 в документе "Руководство по администрированию баз данных ASA" (ASA Database Administration Guide).

☞ Для получения дополнительной информации см. раздел "Параметр GLOBAL_DATABASE_ID" (GLOBAL_DATABASE_ID option) на стр. 569 в документе "Руководство по администрированию баз данных ASA" (ASA Database Administration Guide).

☞ Для получения дальнейшей информации о пулах см. раздел "Использование пулов первичных ключей" на стр. 114.

Использование пулов первичных ключей

Пул первичных ключей представляет собой таблицу, которая содержит набор значений первичных ключей для каждой базы данных в системе SQL Remote. Каждый удаленный пользователь получает свой собственный набор значений первичных ключей. Когда удаленный пользователь вставляет в таблицу новую строку, он использует хранимую процедуру для выбора действительного первичного ключа из пула. Пул поддерживается посредством периодического выполнения в консолидированной базе данных процедуры, которая пополняет запас значений.

Данный способ описывается на примере простой базы данных, в которой хранится информация о торговых представителях и их клиентах. Эти таблицы намного проще по своей структуре тех таблиц, которые использовались бы в реальной базе данных; однако такое упрощение позволяет более подробно рассмотреть аспекты, связанные с репликацией.

Таблица пула первичных ключей

Пул первичных ключей хранится в отдельной таблице. Таблица пула первичных ключей создается с помощью приведенного ниже оператора CREATE TABLE:

```
CREATE TABLE KeyPool ( table_name VARCHAR(40) NOT NULL,  
value INTEGER NOT NULL, location CHAR(12) NOT NULL,  
PRIMARY KEY (table_name, value), );
```

Столбцы этой таблицы имеют следующие значения:

Столбец	Описание
table_name	Содержит имена таблиц, для которых должны храниться пулы первичных ключей. В приведенном ниже простом примере, если бы планировалось добавлять новых торговых представителей только в консолидированной базе данных, пул первичных ключей был бы необходим только для таблицы Customer; поэтому данный столбец можно считать избыточным. Он включен в пример для иллюстрации решения в общем виде.
value	Содержит список значений первичных ключей. Каждое значение уникально для каждой таблицы, перечисленной в столбце table_name.
location	Идентификатор получателя. В некоторых системах этот идентификатор может совпадать со значением rep_key таблицы SalesRep. В других системах помимо торговых представителей могут быть другие пользователи, поэтому эти два идентификатора должны быть отличными друг от друга.

Для повышения производительности можно создать индекс к таблице:

```
CREATE INDEX KeyPoolLocation  
ON KeyPool (table_name, location, value);
```

Репликация пула первичных ключей

Можно либо включать пул ключей в существующую публикацию, либо совместно использовать его как отдельную публикацию. В данном примере для пула первичных ключей создается отдельная публикация.

❖ Процедура репликации пула первичных ключей (SQL)

- 1 Создайте публикацию для данных пула первичных ключей.

```
CREATE PUBLICATION KeyPoolData (
    TABLE KeyPool SUBSCRIBE BY location
);
```

- 2 Создайте подписки на публикацию KeyPoolData для каждой удаленной базы данных.

```
CREATE SUBSCRIPTION
TO KeyPoolData( 'user1' )
FOR user1;

CREATE SUBSCRIPTION
TO KeyPoolData( 'user2' )
FOR user2;
```

...

Аргументом подписки является идентификатор местоположения.

В некоторых случаях имеет смысл добавить таблицу KeyPool к существующей публикации и использовать один и тот же аргумент для создания подписок на каждую публикацию. В данном примере местоположение и значения `rep_key` различны, чтобы показать решение в более общем виде.

☞ Для получения дополнительной информации см. следующие разделы:

- ◆ "Оператор CREATE PUBLICATION" (CREATE PUBLICATION statement) на стр. 314 в документе *"Справочник по SQL для ASA" (ASA SQL Reference Manual)*.
- ◆ "Оператор CREATE SUBSCRIPTION" на стр. 307 данного документа.

Заполнение и пополнение пула ключей

Каждый раз, когда пользователь добавляет информацию о новом клиенте, его пул доступных первичных ключей уменьшается на один ключ. Таблица пула первичных ключей в консолидированной базе данных должна периодически пополняться с помощью следующей процедуры:

```
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
```

```
// убедитесь, имеется 100 значений.  
// Установите верхнее значение согласно  
// конкретным требованиям.  
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;
```

С помощью данной процедуры пул каждого пользователя пополняется до 100 значений. Данное значение устанавливается исходя из того, как часто пользователи вставляют строки в таблицы в базе данных.

Процедуру **ReplenishPool** необходимо периодически запускать в консолидированной базе данных для пополнения пула значений первичных ключей в таблице **KeyPool**.

Для выполнения процедуры **ReplenishPool** необходимо, чтобы для каждого подписчика существовало по крайней мере одно значение первичного ключа, что обеспечит возможность нахождения максимального значения и добавления значения для генерации следующего набора. При первоначальном заполнении пула можно вставить по одному значению для каждого пользователя, после чего вызвать процедуру **ReplenishPool** для заполнения остальных значений. Следующий пример иллюстрирует выполнение данной операции для трех удаленных пользователей и одного консолидированного пользователя по имени **Office**:

```
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();
```

Невозможно использовать триггер для пополнения пула ключей
Для пополнения пула ключей нельзя использовать триггер, так как действия триггера не реплицируются.

Добавление информации о новых клиентах

Если торговому представителю необходимо добавить в таблицу **Customer** информацию о новом клиенте, значение первичного ключа, которое требуется вставить, получается с помощью хранимой процедуры. В данном примере рассматривается хранимая процедура для получения значения первичного ключа, а также хранимая процедура для выполнения оператора **INSERT**.

В данных процедурах используется тот факт, что идентификатором торгового представителя является **CURRENT PUBLISHER** удаленной базы данных.

- ◆ **Процедура NewKey.** Процедура **NewKey** извлекает из пула ключей целочисленное значение и удаляет это значение из пула.

```
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
    // Не используйте последнее значение, так как в
    // этом случае ReplenishPool работать не будет.
    // Во избежание этого пул ключей должен оставаться
    // достаточно наполненным.
    SET @value = NULL;
END IF;
END;

```

- ◆ **Процедура NewCustomer.** Процедура **NewCustomer** вставляет в таблицу информацию о новом клиенте, используя значение, получаемое процедурой **NewKey**, для создания первичного ключа.

```

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

```

Данную процедуру можно улучшить за счет включения проверки значения **new_cust_key**, получаемого из процедуры **NewKey**, на неравенство NULL и, таким образом, предотвратить вставки в случаях, когда это значение - NULL.

Заключительная информация о пулах первичных ключей

Для применения метода с использованием пула первичных ключей требуются следующие компоненты:

- ◆ **Таблица пула ключей.** Таблица, в которой хранятся действительные значения первичных ключей для каждой базы данных в системе.
- ◆ **Процедура пополнения.** Хранимая процедура, которая осуществляет пополнение таблицы пула ключей.
- ◆ **Совместное использование пула ключей.** Каждая база данных в системе должна иметь подписку на свой собственный набор действительных значений из таблицы пула ключей.
- ◆ **Процедуры ввода данных.** Ввод новых строк производится с помощью хранимой процедуры, которая выбирает из пула следующее действительное значение первичного ключа, после чего удаляет его из пула ключей.

Создание подписок

Чтобы подписаться на публикацию, каждому подписчику должны быть предоставлены полномочия удаленного пользователя; также для данного пользователя должна быть создана подписка. Отдельные элементы подписки различаются в зависимости от того, использует ли публикация выражение подписки или нет.

Работа с подписками
в Sybase Central

❖ Создание и управление подписками в Sybase Central

- 1 Откройте папку Publications (эта папка находится в папке SQL Remote).
- 2 Щелкните правой кнопкой мыши по публикации и выберите Properties во всплывающем меню.
- 3 Выберите закладку SQL Remote Subscriptions и сконфигурируйте соответствующие параметры:
 - ◆ Чтобы подписать удаленного пользователя на публикацию, нажмите Subscribe и введите требуемые значения в появившемся диалоге.
 - ◆ Чтобы аннулировать подписку удаленного пользователя, выберите пользователя в списке и нажмите Unsubscribe.
 - ◆ Чтобы вручную активизировать, деактивизировать или синхронизировать подписки, выберите пользователя в списке и нажмите Advanced для открытия диалога Advanced SQL Remote Subscription Actions. В этом диалоге нажмите кнопку Start Now для активизации подписок, кнопку Stop Now для деактивизации и кнопку Synchronize Now для синхронизации подписок.

Изменения в состоянии подписок вступают в действие сразу же после нажатия соответствующей кнопки. Последующее нажатие кнопки Cancel в окне свойств *не* отменяет предыдущего действия по активизации/деактивизации/синхронизации подписок.

Совет

Управлять подписками можно также с помощью закладки Subscriptions в окне свойств удаленного пользователя. Информацию об удаленных пользователях (Remote users) можно найти в двух местах: в папке Users & Groups и в папке Remote Users (в папке SQL Remote).

Подписки без
выражения подписки

Чтобы подписать пользователя на публикацию, в которой нет выражения подписки, необходима следующая информация:

- ◆ **Идентификатор пользователя.** Пользователь, который подписывается на публикацию. Данному пользователю необходимо предоставить полномочия доступа REMOTE.
- ◆ **Имя публикации.** Имя публикации, на которую подписывается пользователь.

Следующий оператор создает подписку для пользователя с идентификатором **SamS** на публикацию **pub_orders_samuel_singer**, которая была создана с использованием раздела WHERE:

```
CREATE SUBSCRIPTION
TO pub_orders_samuel_singer
FOR SamS
```

Подписки с
выражением
подписки

Чтобы подписать пользователя на публикацию, в которой есть выражение подписки, необходима следующая информация:

- ◆ **Идентификатор пользователя.** Пользователь, который подписывается на публикацию. Данному пользователю необходимо предоставить полномочия доступа REMOTE.
- ◆ **Имя публикации.** Имя публикации, на которую подписывается пользователь.
- ◆ **Значение подписки.** Значение, которое должно проверяться выражением подписки публикации. Например, если в публикации есть имя столбца, в котором хранится идентификатор служащего в виде выражения подписки, значение идентификатора подписывающегося пользователя должно быть указано в подписке. Значение подписки всегда является строкой.

Следующий оператор создает подписку для Сэмюэля Сингера (Samuel Singer, идентификатор пользователя **SamS**, идентификатор служащего 856) на публикацию pub_orders, определенную выражением подписки **sales_rep**, которая запрашивает строки с информацией о собственных продажах Сэмюэля Сингера:

```
CREATE SUBSCRIPTION
TO pub_orders ( '856' )
FOR SamS
```

Активизация подписки

Чтобы правильно получать и выполнять обновления, каждый подписчик должен иметь первоначальную копию данных. Процесс синхронизации рассматривается в разделе "Синхронизация баз данных" на стр. 163.

☞ Для получения дополнительной информации см. раздел

- ◆ "Оператор CREATE SUBSCRIPTION" на стр. 307.

Настройка SQL Remote для Adaptive Server Enterprise

Об этой главе

В данной главе описываются принципы настройки инсталляции SQL Remote в случае, когда в качестве консолидированной базы данных используется база данных Adaptive Server Enterprise.

Схожий материал для Adaptive Server Anywhere

Многие принципы проектирования публикаций относятся в равной степени к Adaptive Server Anywhere и Adaptive Server Enterprise, но в отношении команд и возможностей существуют различия. Данная глава во многом совпадает с соответствующей главой, предназначенной для пользователей Adaptive Server Anywhere, "Настройка SQL Remote для Adaptive Server Anywhere" на стр. 77.

Содержание

Раздел	Страница
Общие сведения о настройке	122
Создание публикаций	123
Проектирование публикаций для Adaptive Server Enterprise	127
Разделение таблиц, не содержащих столбца подписки	129
Совместное использование строк в нескольких подписках	136
Управление конфликтами	143
Обеспечение уникальности первичных ключей	151
Создание подписок	156

Общие сведения о настройке

При настройке SQL Remote необходимо решить следующие задачи:

- ◆ **Проектирование публикаций.** Публикации определяют, какая информация совместно используется какими базами данных.
- ◆ **Проектирование подписок.** Подписки определяют, какую информацию получает каждый пользователь.
- ◆ **Реализация проекта.** Создание публикаций и подписок для всех пользователей в системе.

Все функции администрирования выполняются в консолидированной базе данных

Как и все задачи по администрированию SQL Remote, проектирование выполняется администратором базы данных или системным администратором в консолидированной базе данных. Системный администратор Sybase должен выполнять все задачи по конфигурированию SQL Remote.

☞ Для получения дополнительной информации о среде Adaptive Server Enterprise см. документацию по Adaptive Server Enterprise.

Создание публикаций

В этом разделе

В данном разделе описывается создание простых публикаций, состоящих из целых таблиц или из постолбцовых подмножеств таблиц.

☞ Простые публикации также рассматриваются в главе "Учебный раздел для пользователей Adaptive Server Enterprise" на стр. 47.

Создание статей из целых таблиц

Самая простая статья состоит из одной статьи, которая включает в себя все строки и столбцы одной или более таблиц.

❖ Процедура создания статьи, в которую входят все строки и столбцы таблицы

- 1 Отметьте таблицу для репликации. Для этого выполните процедуру **sp_add_remote_table**:

```
sp_add_remote_table имя-таблицы
```

- 2 Добавьте таблицу в публикацию. Для этого выполните процедуру **sp_add_article**:

```
sp_add_article имя-публикации, имя-таблицы
```

Пример

- ◆ С помощью следующих команд таблица **SalesRep** добавляется в публикацию **SalesRepData**:

```
sp_add_remote_table 'SalesRep'
sp_add_article 'SalesRepData', 'SalesRep'
go
```

Создание статей, содержащих несколько столбцов в таблице

Для создания статьи, включающей только некоторые столбцы из таблицы, необходимо перечислить столбцы, которые необходимо включить в статью, с помощью процедуры **sp_add_article_col**. При отсутствии списка столбцов статья будет содержать все столбцы таблицы.

❖ Процедура создания статьи, содержащей некоторые столбцы и все строки таблицы

- 1 Отметьте таблицу для репликации. Для этого выполните процедуру **sp_add_remote_table**:

```
sp_add_remote_table имя-таблицы
go
```

- 2 Добавьте таблицу в публикацию. Для этого выполните процедуру **sp_add_article**:

```
sp_add_article имя-публикации, имя-таблицы
go
```

Процедура **sp_add_article** добавляет таблицу в публикацию. По умолчанию в публикацию добавляются все столбцы таблицы. Если требуется добавлять только некоторые столбцы, необходимо использовать процедуру **sp_add_article_col** для указания того, какие столбцы нужно включить.

- 3 Добавьте в публикацию отдельные столбцы. Для этого выполните процедуру **sp_add_article_col** для каждого столбца:

```
sp_add_article_col имя-публикации, имя-таблицы, имя-
столбца
go
```

Пример

- ◆ При помощи следующей команды добавляется только столбец **rep_key** таблицы SalesRep в публикацию SalesRepData:

```
sp_add_remote_table 'SalesRep'
sp_add_article 'SalesRepData', 'SalesRep'
sp_add_article_col 'SalesRepData', 'SalesRep', 'rep_key'
go
```

Создание статей, содержащих несколько строк в таблице

Существует два способа включения в статью только некоторых строк из таблицы:

- ◆ **Раздел WHERE.** Для включения в статью поднабора строк можно использовать раздел WHERE. Все подписчики на публикацию, содержащую данную статью, получают строки, которые удовлетворяют условию раздела WHERE.
- ◆ **Столбец подписки.** Для включения любого другого набора строк в разные подписки на публикации, содержащие данную статью, можно использовать столбец подписки.

Разрешенные разделы

В SQL Remote для Adaptive Server Enterprise на каждый из приведенных выше случаев налагаются следующие ограничения:

- ◆ **Ограничения раздела WHERE.** Поддерживается только одно выражение раздела WHERE, а именно следующее:

```
WHERE имя-столбца IS NOT NULL.
```

- ◆ **Столбец подписки.** SQL Remote для Adaptive Server Anywhere кроме имен столбцов поддерживает и другие выражения. Для Adaptive Server Enterprise выражение подписки должно быть представлено именем столбца.

Случаи использования разделов WHERE и SUBSCRIBE BY

Выражение подписки необходимо использовать тогда, когда разные подписчики на публикацию должны получать разные строки из таблицы. Выражение подписки является самым мощным инструментом разделения таблиц.

Создание статьи с использованием раздела WHERE

Раздел WHERE используется для исключения набора строк из всех подписок на публикацию.

❖ Процедура создания статьи с использованием раздела WHERE

- 1 Отметьте таблицу для репликации, если это еще не было сделано. Для этого выполните процедуру **sp_add_remote_table**:

```
sp_add_remote_table имя_таблицы
```

- 2 Добавьте таблицу в публикацию. Для этого выполните процедуру **sp_add_article**. Укажите имя столбца, соответствующее разделу **WHERE столбец IS NOT NULL** в третьем аргументе в процедуре:

```
sp_add_article имя_публикации,
            имя_таблицы,
            имя_столбца
```

Не указывайте **IS NOT NULL**; это подразумевается. Необходимо указать только имя столбца.

- 3 Если необходимо включить только поднабор столбцов из таблицы, укажите соответствующие столбцы с помощью процедуры **sp_add_article_col**. В статью необходимо включать столбцы, указанные в соответствующем разделе WHERE.

Пример

- ◆ С помощью следующего набора операторов создается публикация, содержащая единственную статью, в которую входят только те строки **test_table**, для которых в столбце **col_1** значение не равно нулю:

```
sp_create_publication test_pub
sp_add_remote_table test_table
sp_add_article test_pub, test_table, col_1
go
```

Создание статьи с использованием столбца подписки

Столбец подписки используется тогда, когда необходимо, чтобы строки совместно использовались многими удаленными базами данных.

❖ Процедура создания статьи с использованием столбца подписки

- 1 Отметьте таблицу для репликации, если это еще не было сделано. Для этого выполните процедуру **sp_add_remote_table**:

```
sp_add_remote_table имя_таблицы
```

- 2 Добавьте таблицу в публикацию. Для этого выполните процедуру **sp_add_article**. Укажите имя столбца, который требуется использовать как выражение подписки, в четвертом аргументе в процедуре:

```
sp_add_article имя_публикации, имя_таблицы, NULL,
            имя_столбца
```

Ввод записи NULL позволяет избежать добавления раздела WHERE.

- 3 Если в таблицу необходимо включить только поднабор столбцов, укажите требуемые столбцы с помощью процедуры **sp_add_article_col**. В статью также следует включать столбец, указанный в выражении подписки.

Пример

- ◆ С помощью приведенного ниже набора операторов создается публикация, включающая одну статью, которая поддерживает подписки на основе значения столбца **col_1**:

```
sp_create_publication test_pub
sp_add_remote_table test_table
sp_add_article test_pub,
            test_table,
            NULL,
            col_1
go
```

Примечания относительно статей

- ◆ В статье можно объединять раздел WHERE и выражение подписки.
- ◆ В статью должны быть включены все столбцы в первичном ключе.
- ◆ Не следует включать в статью поднабор столбцов, за исключением следующих случаев:
 - ◆ Остающиеся столбцы имеют значения по умолчанию или допускают значение NULL.

- ◆ В удаленных базах данных вставки не выполняются. Обновления не вызывали бы проблем, если бы они не изменяли значения первичного ключа.

В остальных ситуациях, отличных от тех, что описаны выше, при включении в статью подбора столбцов операторы INSERT в консолидированной базе данных выполняться не будут.

Проектирование публикаций для Adaptive Server Enterprise

Ознакомившись с процедурой создания простых публикаций, рассмотрим вопрос правильного проектирования публикаций. В данном разделе рассматриваются различные аспекты проектирования публикаций и описываются методы правильного проектирования.

Принципы настройки

Каждая подписка должна быть полной реляционной базой данных

Удаленная база данных использует информацию в своих подписках совместно с консолидированной базой данных. Подписка одновременно является поднабором реляционной базы данных, хранящейся на консолидированном узле, и полной реляционной базой данных на удаленном узле. Поэтому информация в подписке подчиняется тем же правилам, что и любая другая реляционная база данных:

- ◆ **Должны быть действительны связи по внешнему ключу.** Для каждой записи во внешнем ключе должна существовать соответствующая запись первичного ключа в базе данных.

Утилита извлечения базы данных производит проверку того, что операторы CREATE TABLE для удаленных баз данных не имеют внешних ключей, определенных для не существующих в удаленной базе данных таблиц.

- ◆ **Должна сохраняться уникальность первичного ключа.** Невозможно проверить то, какие новые строки были введены на других узлах, но еще не были реплицированы. При проектировании должна быть исключена возможность добавления пользователями на различных узлах строк с идентичными значениями первичного ключа, поскольку это может привести к конфликтам при репликации строк в консолидированную базу данных.

Целостность транзакций при отсутствии блокирования должна сохраняться

Данные в рассредоточенной базе данных (которая состоит из консолидированной базы данных и всех удаленных баз данных) должны сохранять свою целостность при выполнении обновлений на всех узлах, даже если не существует механизма блокирования в масштабе всей системы для любой отдельной строки.

- ◆ **Должны предотвращаться или разрешаться конфликты блокирования.** В системе SQL Remote не существует способа блокирования строк во всех базах данных для предотвращения одновременного изменения строк различными пользователями. Подобные конфликты необходимо предотвращать путем выработки соответствующих решений за пределами системы, либо решать их должным образом в консолидированной базе данных.

Эти основные особенности реляционных баз данных должны быть учтены при разработке публикаций и подписок. В данной главе рассматриваются принципы и методы правильного проектирования.

Условия создания действительных статей

Поддержка операторов INSERT в удаленных базах данных

В статью должны быть включены все столбцы в первичном ключе.

Чтобы операторы INSERT в удаленной базе данных могли создавать точные копии в консолидированной базе данных, можно исключать из статьи только те столбцы, которые не обязательно должны присутствовать в действующем операторе INSERT. Таковыми являются:

- ◆ Столбцы, которые допускают значение NULL.

- ◆ Столбцы, которые имеют значения по умолчанию.

Если исключить какой-либо столбец, не отвечающий одному из этих требований, операторы INSERT, выполняемые в удаленной базе данных, не будут работать после их репликации в консолидированную базу данных.

<p>Консолидированная база данных INSERT INTO SalesRep (ID, Rep) VALUES (3, 'Shih')</p>	<table border="1"> <thead> <tr> <th>ID</th> <th>Rep</th> <th>Dept</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Ann</td> <td>101</td> </tr> <tr> <td>2</td> <td>Marc</td> <td>101</td> </tr> <tr> <td>3</td> <td>Shih</td> <td>X</td> </tr> </tbody> </table>	ID	Rep	Dept	1	Ann	101	2	Marc	101	3	Shih	X	<p>Оператор Insert не выполняется</p>
ID	Rep	Dept												
1	Ann	101												
2	Marc	101												
3	Shih	X												
<p>Удаленная база данных INSERT INTO SalesRep (ID, Rep) VALUES (3, 'Shih')</p>	<table border="1"> <thead> <tr> <th>ID</th> <th>Rep</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Ann</td> </tr> <tr> <td>2</td> <td>Marc</td> </tr> <tr> <td>3</td> <td>Shih</td> </tr> </tbody> </table>	ID	Rep	1	Ann	2	Marc	3	Shih	<p>Оператор Insert успешно выполняется</p>				
ID	Rep													
1	Ann													
2	Marc													
3	Shih													

Условия включения отдельных строк

Существует два способа включения в публикацию только некоторых строк:

- ◆ **Раздел WHERE.** Для включения в статью поднабора строк можно использовать раздел WHERE. Все подписчики на публикацию, содержащую эту статью, будут получать строки, которые удовлетворяют условию раздела WHERE.

В SQL Remote для Adaptive Server Enterprise поддерживается только один раздел WHERE, а именно следующий:

WHERE имя_столбца IS NOT NULL

- ◆ **Столбцы подписки.** Для включения любого другого набора строк в разные подписки на публикации, содержащие данную статью, можно использовать столбец подписки.

☞ Для получения дополнительной информации об ограничениях на строки см. раздел "Создание статей, содержащих несколько строк в таблице" на стр. 124.

Разделение таблиц, не содержащих столбца подписки

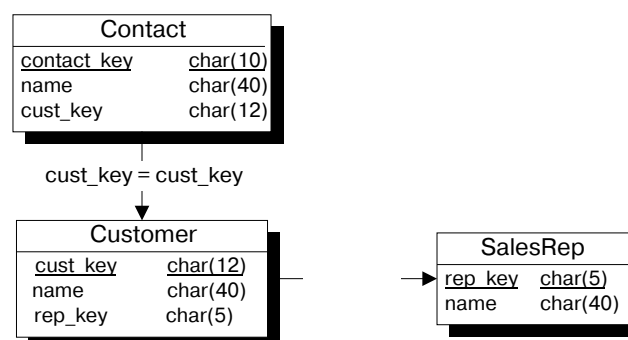
Довольно часто возникает необходимость разделить строки таблицы, даже когда в таблице отсутствует столбец подписки. В данном разделе приводится пример метода разрешения этой проблемы.

Пример базы данных Contact

На примере базы данных Contact иллюстрируются принципы и методы разделения таблиц, не содержащих столбец подписки.

Пример

Ниже в качестве примера представлена простая база данных. Эта база данных называется Contact, так как в дополнение к двум другим таблицам, описанным ранее в этой главе, содержит таблицу контактов (Contact).



Каждый торговый представитель ведет работу с несколькими клиентами. У некоторых клиентов есть одно контактное лицо, в то время как у других клиентов есть несколько контактных лиц.

Таблицы в базе данных

Ниже приведено более подробное описание этих трех таблиц:

Таблица	Описание
SalesRep	<p>Все торговые представители, работающие в компании. Таблица SalesRep имеет следующие столбцы:</p> <ul style="list-style-type: none"> ◆ rep_key - идентификатор торгового представителя. Этот столбец является первичным ключом. ◆ name - имя торгового представителя. <p>Эта таблица создается с помощью следующего оператора SQL:</p> <pre> CREATE TABLE SalesRep (rep_key CHAR(12) NOT NULL, name CHAR(40) NOT NULL, PRIMARY KEY (rep_key)) go </pre>

Таблица	Описание
Customer	<p>Все клиенты, ведущие дела с данной компанией. Таблица Customer состоит из следующих столбцов:</p> <ul style="list-style-type: none"> ◆ cust_key - идентификатор клиента. Этот столбец является первичным ключом. ◆ name - имя клиента. ◆ rep_key - идентификатор торгового представителя, который работает с данным клиентом. Этот столбец является внешним ключом к таблице SalesRep. <p>Эта таблица создается с помощью следующего оператора SQL:</p> <pre> 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 </pre>
Contact	<p>Все отдельные контактные лица, которые имеют дело с компанией. Каждый контакт принадлежит отдельному клиенту. Таблица Contact состоит из следующих столбцов:</p> <ul style="list-style-type: none"> ◆ contact_key - идентификатор контактного лица. Этот столбец является первичным ключом. ◆ name - имя контакта. ◆ cust_key - идентификатор клиента, с которым соотносится данное контактное лицо. Этот столбец является внешним ключом к таблице Customer. <p>Эта таблица создается с помощью следующего оператора SQL:</p> <pre> 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 </pre>

Цели репликации

Цель создаваемого проекта репликации данных заключается в обеспечении каждого торгового представителя следующей информацией:

- ◆ Полная таблица SalesRep;
- ◆ Информация о закрепленных за торговыми представителями клиентах из таблицы Customer;
- ◆ Информация о контактах, закрепленных за соответствующими клиентами, из таблицы Contact;
- ◆ Сохранение достоверности информации при перераспределении областей торговых представителей.

Перераспределение областей в примере базы данных Contact

При **перераспределении областей** происходит переназначение строк между подписчиками. В данном случае перераспределение областей подразумевает под собой переназначение клиентов между торговыми представителями. Это перераспределение выполняется путем обновления столбца **rep_key** таблицы **Customer**.

Оператор UPDATE реплицируется как INSERT или DELETE соответственно для прежнего и нового торгового представителя для того, чтобы строка с информацией о клиенте была правильно перемещена к новому торговому представителю.

Отсутствие записей в журнале для таблицы Contact при перераспределении областей

При переназначении клиента изменения в таблицу **Contact** не вносятся. Никаких изменений в таблице **Contact** не происходит, поэтому никакие записи относительно таблицы **Contact** в журнал транзакций не добавляются. Из-за отсутствия информации в журнале SQL Remote не может переназначать строки таблиц **Contact** и **Customer**. Это приводит к возникновению проблем с ссылочной целостностью: таблица **Contact** в удаленной базе данных прежнего торгового представителя содержит значение **cust_key**, для которого **Customer** отсутствует.

В этом разделе описывается способ переназначения строк таблицы **Contact**.

Разделение таблицы Customer в примере базы данных Contact

Таблица **Customer** может быть разделена с использованием значения **rep_key** в качестве столбца подписки. Публикация, которая включает таблицы **SalesRep** и **Customer**, создается следующим образом:

```
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
```

Добавление столбца списка подписки в таблицу Contact

Таблица **Contact** должна быть также разделена между торговыми представителями, однако она не содержит ссылок на значение **rep_key** торговых представителей.

Добавление столбца списка подписки

Для решения этой проблемы в Adaptive Server Enterprise необходимо добавить столбец в таблицу **Contact**, который содержит список разделенных запятыми значений подписки на строку. (В данном случае можно ввести только одно значение подписки) Обработку столбца можно выполнять с использованием триггеров, чтобы наличие этого столбца не влияло на приложения, работающие с базой данных. Такой столбец называется **столбцом списка подписки**.

При вставке, обновлении или удалении строки в таблице **Customer** триггер обновляет строки в таблице **Contact**. В частности, триггер обновляет столбец списка подписки. Поскольку таблица **Contact** отмечена для репликации, в журнал заносится образ строки до и после обновления.

В журнал заносятся значения, а не подписчики
 Хотя в данном случае введенные значения представляют собой информацию о подписчиках, список подписчиков в журнал не вносится. Сервер обрабатывает только информацию о публикациях, а Message Agent работает со всей информацией о подписчиках. Значения, вводимые в журнал, предназначены для сравнения со значением подписки в каждой подписке. Например, если бы строки таблицы были разделены между торговыми представителями по стране или региону, в журнал транзакций вносилось бы значение страны или региона.

Столбец списка подписки – это столбец, добавляемый в таблицу исключительно для хранения списка разделенных запятыми подписчиков. В данном случае может быть только один единственный подписчик на каждую строку таблицы **Contact**, поэтому столбец списка подписки содержит только одно значение.

☞ Описание случая, когда столбец списка подписки может содержать много значений, см. в разделе "Совместное использование строк в нескольких подписках" на стр. 158.

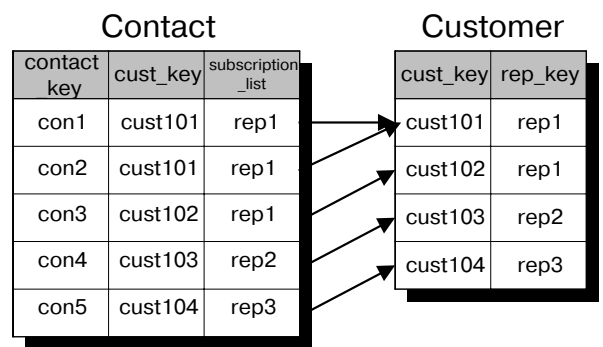
Определение таблицы Contact

В случае с таблицей Contact определение таблицы было бы изменено следующим образом:

```
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
```

Создается дополнительный столбец, допускающий значения NULL, чтобы существующие приложения могли продолжать работать с базой данных без изменений.

Столбец списка подписки **subscription_list** содержит значение **rep_key**, которое соответствует строке со значением первичного ключа **cust_key** в таблице Customer. Обработку и обновление информации в столбце **subscription_list** выполняет набор триггеров.



☞ В консолидированной базе данных Adaptive Server Enterprise используется другое решение данной проблемы. Для получения дополнительной информации см. раздел "Разделение таблиц, не содержащих выражение подписки" на стр. 90.

Обработка данных столбца списка подписки

Для обеспечения достоверности данных в столбце списка подписки **subscription_list** необходимо использовать триггеры в следующих операциях:

- ◆ INSERT с таблицей Contact;
- ◆ UPDATE с таблицей Contact;
- ◆ UPDATE с таблицей Customer.

Операция UPDATE с таблицей Customer решает проблему **перераспределения областей**, когда клиенты переназначаются другим торговым представителям.

Триггер INSERT
таблицы Contact

Триггер для выполнения операции INSERT с таблицей Contact устанавливает значение **subscription_list** в значение **rep_key** из таблицы Customer:

```
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
```

Триггер обновляет столбец **subscription_list** для вставляемых строк; эти строки устанавливаются следующим подзапросом:

```
SELECT contact_key
FROM inserted
```

Триггер UPDATE
таблицы Contact

Триггер для операции UPDATE с таблицей Contact проверяет, изменяется ли столбец **cust_key**, и если он изменился, обновляет столбец **subscription_list**.

```
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
```

Триггер написан с использованием оператора join; можно было бы использовать и подзапрос.

Триггер UPDATE для
таблицы Customer

Следующие триггеры управляют обновлением информации о клиентах, перемещая ее к новым торговым представителям:

```
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
```

Повышение производительности при операции извлечения

При извлечении или синхронизации информации о пользователе, использование *списка подписки* может снизить производительность, поскольку при этом возникает необходимость сканирования всей таблицы.

При извлечении баз данных для большого количества пользователей и возникновении проблем с производительностью для улучшения производительности можно использовать **представление подписки**. Это представление должно содержать подзапрос, который используется только для извлечения и синхронизации и игнорируется во время сканирования журнала. Упомянутые таблицы, тем не менее, должны содержать триггеры для обработки данных столбца *списка подписки*.

❖ Процедура создания представления подписки

- 1 Постройте запрос, который будет использовать подзапрос для выбора нужных для подписки строк из таблицы.

Например, продолжая пример из предыдущих разделов, приведенный ниже запрос выбирает строки таблицы *Contact* для подписчика со значением **rep5** в *rep_key*:

```
SELECT * FROM Contact
WHERE 'rep5' = (SELECT rep_key
               FROM Customer
               WHERE cust_key = Contact.cust_key )
```

- 2 Создайте представление, которое содержит этот подзапрос. Например:

```
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 )
```

В этом определении представления не важно, какое значение используется в левой части раздела *WHERE* (*repxx* в описанном выше примере). Инструментальные средства репликации используют подзапрос только для извлечения и синхронизации. Извлекаются и синхронизируются строки, для которых значение *SUBSCRIBE BY* равно результирующему набору подзапроса.

- 3 Задайте имя представления в качестве параметра для *sp_add_article* или *sp_modify_article*:

```
exec sp_add_remote_table 'Contact'
go
exec sp_add_article SalesRepData,
                   'Contact',
                   NULL,
                   'subscription_list',
                   'Contact_sub_view'
```

Столбец *subscription_list* используется для сканирования журнала, а подзапрос - для извлечения и синхронизации.

☞ Для получения дополнительной информации см. разделы "Повышение производительности при операции извлечения совместно используемых строк" на стр. 140, "Процедура sp_add_article" на стр. 331 и "Процедура sp_modify_article" на стр. 346.

Совместное использование строк в нескольких подписках

В некоторых случаях бывает необходимо включить строку в несколько подписок. Например, это относится к случаям, когда вместо связи "один ко многим", которая рассматривалась ранее, между клиентами и торговыми представителями существует связь "многие ко многим".

Пример базы данных Policy

Пример с использованием базы данных Policy иллюстрирует, зачем и как разделять таблицы при наличии в базе данных связи "многие ко многим".

Пример базы данных Здесь в целях иллюстрации приводится простая база данных.

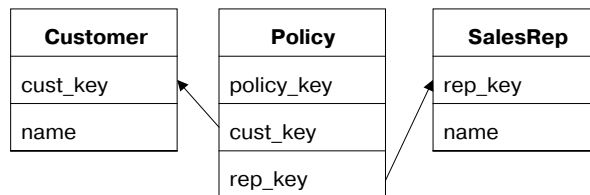


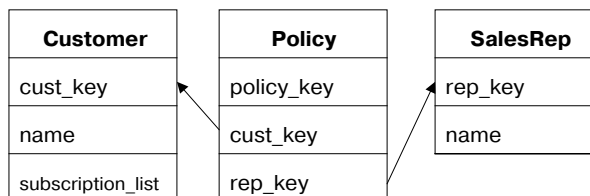
Таблица Policy имеет по одной строке для каждого набора страховых полисов. Каждый полис составляется для клиента отдельным торговым представителем. Между клиентами и торговыми представителями существует связь "многие ко многим", а каждой отдельной паре "представитель-клиент" может соответствовать несколько полисов.

Может потребоваться, чтобы любая строка в таблице Customer совместно использовалась одним или несколькими торговыми представителями, либо не использовалась ими вообще.

Решение проблемы

Для решения подобной ситуации требуется написать триггеры, которые создавали бы список разделенных запятыми значений для хранения в избыточном столбце списка подписки в таблице Customer и включали бы этот столбец как столбец подписки при добавлении таблицы Customer в публикацию. Строка совместно используется любой подпиской, для которой значение подписки совпадает с любым значением в столбце списка подписки.

База данных со включенным столбцом списка подписки имеет следующий вид:



Столбцы VARCHAR в Adaptive Server Enterprise имеют ограничение на ввод не более 255 символов, что ограничивает число значений, которые можно хранить в списке разделенных запятыми значений.

Определения таблиц Определения таблиц выглядят следующим образом:

```

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)
)

```

Примечания

♦ Столбец **subscription_list** в таблице Customer может принимать значения NULL, с тем чтобы в столбец **subscription_list** можно было добавлять информацию о клиентах, которые не имеют торговых представителей.

Публикация

Для создания публикации для этой базы данных можно использовать следующий набор операторов:

```

// Выбор таблиц для репликации
exec sp_add_remote_table 'SalesRep'
exec sp_add_remote_table 'Policy'
exec sp_add_remote_table 'Customer'
go

// Создание пустой публикации
exec sp_create_publication 'SalesRepData'

// Добавление таблицы Sales Rep в публикацию
exec sp_add_article 'SalesRepData', 'SalesRep'

// Добавление таблицы Policy в публикацию
exec sp_add_article 'SalesRepData', 'Policy', NULL,
'rep_key'

// Добавление таблицы Customer в публикацию
// Подписка по столбцу subscription_list
// Исключение столбца subscription_list
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

```

Подписки на эту публикацию принимают следующую форму:

```

exec sp_subscription 'create',
'SalesRepData',
'userID',
'rep_key'
go

```

Где *userID* идентифицирует подписчика, а *rep_key* - столбец подписки, в котором хранятся значения столбца **rep_key** из таблицы **SalesRep**.

Обработка данных в столбце списка подписки

Для обработки данных в столбце списка подписки, включенного в таблицу Customer, необходимо написать процедуру и набор триггеров. Этот раздел посвящен рассмотрению данных объектов.

Хранимая процедура Следующая процедура используется для создания столбца списка подписки и вызывается из триггеров, которые обрабатывают данные в столбце subscription_list.

```
CREATE PROCEDURE SubscribeCustomer @cust_key CHAR(12)
AS
BEGIN
-- Rep возвращает список торговых представителей
-- для клиента @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)

-- создание списка значений rep_key через запяты
-- для этого клиента
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

-- Обновление subscription_list в таблице Customer
UPDATE Customer
SET subscription_list = @subscription_list
WHERE cust_key = @cust_key
END
```

Примечания

- ◆ Процедура принимает ключ Customer в качестве входного аргумента.
- ◆ Rep является курсором для запроса, который вносит в список каждого торгового представителя, с которым клиент заключил договор.
- ◆ Цикл WHILE создает переменную VARCHAR (255), в которой хранится список торговых представителей, перечисленных через запятую.

Триггеры Приведенный ниже триггер обновляет столбец **subscription_list** таблицы Customer после вставки строки в таблицу Policy.

```
CREATE TRIGGER InsPolicy
ON Policy
FOR INSERT
AS
BEGIN
-- Cust возвращает информацию о введенных
-- клиентах
DECLARE Cust CURSOR FOR
    SELECT DISTINCT cust_key
    FROM inserted
DECLARE @cust_key CHAR(12)
OPEN Cust
-- Обновление списка торговых представителей новым
```

```

-- значением rep для каждого клиента
WHILE 1=1 BEGIN
    FETCH Cust INTO @cust_key
    IF @@sqlstatus != 0 BREAK
    EXEC SubscribeCustomer @cust_key
END
END

```

Следующий триггер обновляет столбец **subscription_list** таблицы Customer, когда из таблицы Policy удаляется строка.

```

CREATE TRIGGER DelPolicy
ON Policy
FOR DELETE
AS
BEGIN
    -- Cust возвращает информацию об удаленных
    -- клиентах
    DECLARE Cust CURSOR FOR
        SELECT DISTINCT cust_key
        FROM deleted
    DECLARE @cust_key CHAR(12)

    OPEN Cust
    -- Обновление списка торговых представителей для
    -- каждого клиента, теряющего представителя
    WHILE 1=1 BEGIN
        FETCH Cust INTO @cust_key
        IF @@sqlstatus != 0 BREAK
        EXEC SubscribeCustomer @cust_key
    END
END

```

Исключение столбца списка подписки из публикации

Необходимо исключить из публикации столбец списка подписки, поскольку включение этого столбца приведет к репликации чрезмерного числа обновлений.

Для примера рассмотрим случай, когда существует много полисов для каждого клиента. При назначении клиенту нового торгового представителя запускается триггер для обновления столбца списка подписки в таблице Customer. Если столбец списка подписки является частью публикации, будет реплицироваться одно обновление для каждого полиса всем торговым представителям, закрепленным за этим клиентом.

Триггеры, используемые только в консолидированной базе данных

Значения в столбце списка подписки обрабатываются с помощью триггеров. Эти триггеры запускаются в консолидированной базе данных при выполнении вставок или обновлений в Message Agent. Необходимо исключить триггеры из удаленных баз данных, так как они обрабатывают данные в несуществующем столбце.

Для извлечения из удаленных баз данных только определенных триггеров можно использовать процедуру **sp_user_extraction_hook**. Эта процедура вызывается как завершающий этап операции извлечения. По умолчанию эта процедура является пустой.

❖ Настройка процедуры извлечения для исключения определенных триггеров

- 1 Убедитесь, что параметр **quoted_identifier** установлен в значение ON:

```

set quoted_identifier on
go

```

- 2 Необходимо обеспечить наличие всех временных таблиц, на которые ссылается процедура, так как в противном случае оператор CREATE PROCEDURE выполняться не будет. Временные таблицы, на которые ссылается следующая процедура, доступны в сценарии *ssremote.sql*. Скопируйте из сценария все необходимые определения таблицы и перед созданием процедуры выполните операторы CREATE TABLE для обеспечения их наличия в текущем сеансе подключения.

3 Создайте следующую процедуру:

```
CREATE PROCEDURE sp_user_extraction_hook
AS
BEGIN
    -- Нет необходимости извлекать триггеры INSERT и
    -- DELETE, созданные в таблице Policy, для обработки
    -- данных столбца subscription_list, поскольку этот
    -- столбец не включается в публикацию.
    -- Если бы эти объекты были извлечены,
    -- триггеры INSERT
    -- не выполнялись бы на удаленной базе данных,
    -- поскольку ссылаются на столбец
    -- (subscription_list), который не существует.
    DELETE FROM #systrigger
    WHERE table_id = object_id( 'Policy' )
    -- Не создавать процедур
    DELETE FROM #sysprocedure
    WHERE proc_name = 'SubscribeCustomer'
END
go
```

Повышение производительности при операции извлечения совместно используемых строк

При извлечении или синхронизации информации о пользователе наличие столбца *списка подписки* может привести к снижению производительности, так как необходимо будет выполнить сканирование всей таблицы.

При извлечении баз данных для большого количества пользователей и возникновении проблем с производительностью для улучшения производительности можно использовать **представление подписки**. Это представление должно содержать подзапрос, который используется только для извлечения и синхронизации и игнорируется во время сканирования журнала. Упомянутые таблицы, тем не менее, должны содержать триггеры для обработки данных столбца *списка подписки*.

❖ Процедура создания представления подписки

- 1 Постройте запрос, который будет использовать подзапрос для выбора нужных для подписки строк из таблицы.

Например, продолжая пример из предыдущих разделов, приведенный ниже запрос выбирает строки таблицы *Contact* для подписчика со значением **rep5** в *rep_key*:

```
SELECT * FROM Contact
WHERE 'rep5' = (SELECT rep_key
                FROM Customer
                WHERE cust_key = Contact.cust_key )
```

- 2 Создайте представление, которое содержит этот подзапрос. Например:

```
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
                  )
```

В этом определении представления не важно, какое значение используется в левой части раздела WHERE (*repxx* в описанном выше примере). Инструментальные средства репликации используют подзапрос только для извлечения и синхронизации. Извлекаются и синхронизируются строки, для

которых значение SUBSCRIBE BY равно результирующему набору подзапроса.

- 3 Задайте имя представления в качестве параметра для `sp_add_article` или `sp_modify_article`:

```
exec sp_add_article SalesRepData,
    'Customer',
    NULL,
    'subscription_list',
    'Customer_sub_view'
```

Столбец `subscription_list` используется для сканирования журнала, а подзапрос - для извлечения и синхронизации.

☞ Для получения дополнительной информации см. разделы "Повышение производительности при операции извлечения" на стр. 134, "Процедура `sp_add_article`" на стр. 331 и "Процедура `sp_modify_article`" на стр. 346.

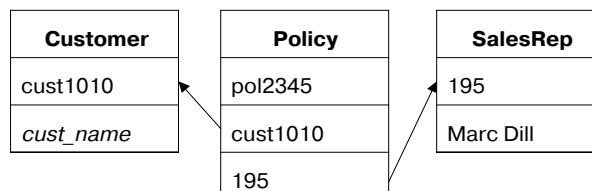
Использование параметра `Subscribe_by_remote` со связями "многие ко многим"

Когда параметр `SUBSCRIBE_BY_REMOTE` установлен в значение `ON`, при выполнении операций с удаленных баз данных над строками с нулевым значением параметра `SUBSCRIBE BY` или же с пустой строкой считается, что удаленный пользователь подписан на данную строку. По умолчанию параметр `SUBSCRIBE_BY_REMOTE` установлен в `ON`. В большинстве случаев рекомендуется использовать именно это значение.

С помощью параметра `SUBSCRIBE_BY_REMOTE` решается проблема, которая в противном случае возникла бы с некоторыми публикациями, в том числе и с таблицей `Policy` из предыдущего примера. В данном разделе приводится описание данной проблемы и способа автоматического предотвращения ее возникновения с помощью данного параметра.

В базе данных используется столбец списка подписки для адресации к таблице `Customer`, поскольку каждый клиент (из таблицы `Customer`) может быть связан с несколькими торговыми представителями (из таблицы `SalesReps`).

Марк Дилл (Marc Dill) - торговый представитель, который только что начал работать с новым клиентом. Он вставляет новую строку в таблицу **Customer** и еще одну строку в таблицу **Policy**, чтобы закрепить за собой нового клиента. Если допустить, что столбец списка подписки не включен в публикацию, в удаленной базе данных Марка Дилла будут выполняться следующие операции:



Поскольку в консолидированной базе данных операцию `INSERT` со строкой в таблице `Customer` выполняет `Message Agent`, при выполнении операции `INSERT` `Adaptive Server Enterprise` заносит значение подписки в журнал транзакций.

Позже, когда `Message Agent` сканирует журнал, он формирует список подписчиков на новую строку, используя значение подписки, хранящееся в журнале; информация о Марке Дилле в этом списке отсутствует. Если бы параметр `Subscribe_by_remote` был установлен в значение `OFF`, информация о новом клиенте отсылалась бы назад к Марку Диллу как результат выполнения операции `DELETE`.

Если параметр SUBSCRIBE_BY_REMOTE установлен в значение ON, Message Agent считает, что, поскольку столбец списка подписки равен NULL, строка принадлежит тому торговому представителю, который внес ее в таблицу. В результате INSERT не реплицируется обратно к Марку Диллу, и данные в системе репликации сохраняют согласованность.

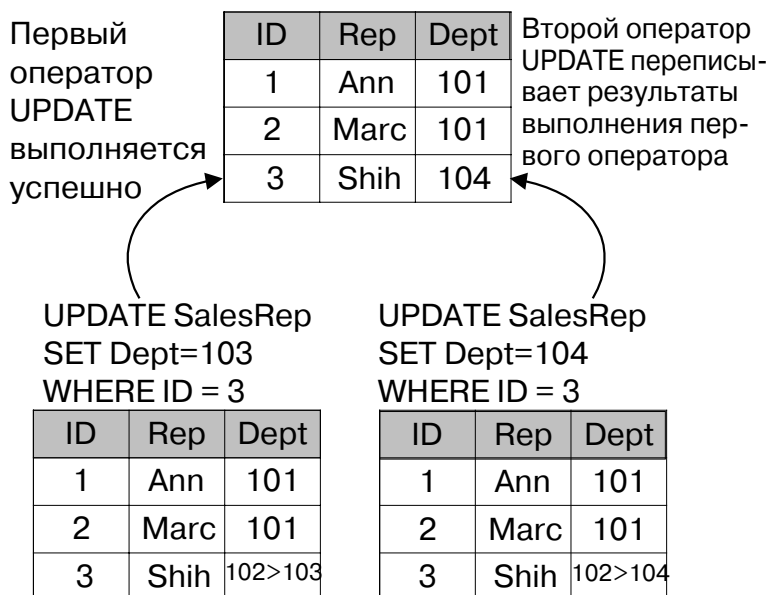
Для работы со столбцом списка подписки можно использовать триггер, который выполняется после операции INSERT.

Управление конфликтами

Конфликт при выполнении операций обновления (UPDATE) происходит при возникновении следующей последовательности событий:

- 1 Пользователь 1 обновляет строку на удаленном узле 1.
- 2 Пользователь 2 обновляет ту же самую строку на удаленном узле 2.
- 3 Обновление от пользователя 1 реплицируется в консолидированную базу данных.
- 4 Обновление от пользователя 2 реплицируется в консолидированную базу данных.

При репликации операторов UPDATE в SQL Remote Message Agent каждый оператор UPDATE выполняется отдельно для каждой строки. Кроме того, сообщение также содержит старые значения строки для сравнения. Когда консолидированная база данных получает обновление от пользователя 2, значения в строке не совпадают с теми, что записаны в сообщении.



Разрешение конфликтов по умолчанию

По умолчанию выполнение UPDATE все же продолжается, с тем, чтобы обновление от пользователя 2 (последнее обновление, поступившее в консолидированную базу данных) было внесено в консолидированную базу данных и реплицировано во все остальные базы данных пользователей, подписанных на данную строку. В общем случае применяемый по умолчанию метод решения конфликта заключается в выполнении более поздней операции (в нашем случае это операция обновления от пользователя 2), при этом конфликт не регистрируется. Обновление от пользователя 1 не сохраняется.

SQL Remote также позволяет применять другие методы разрешения конфликтов с использованием для этих целей триггера, который дает возможность изменять данные нужным образом.

Конфликты не возникают из-за обновления первичного ключа

Конфликты операторов UPDATE *не* возникают в связи с обновлением первичного ключа. Если обновляемый столбец содержит первичные ключи, то, когда в консолидированную базу данных поступает обновление от пользователя 2, обновление строк не выполняется.

В данном разделе описываются приемы разрешения конфликтов в консолидированной базе данных в инсталляции SQL Remote.

Принципы управления конфликтами в SQL Remote

При обнаружении конфликта

Сообщения репликации SQL Remote включают в себя операторы UPDATE в виде набора обновлений отдельных строк, при этом каждый оператор имеет раздел VERIFY, который содержит значения до обновления.

Конфликт обновлений фиксируется сервером базы данных в случае, когда не удастся сопоставить значения раздела VERIFY строкам в базе данных.

Обнаружение и разрешение конфликтов выполняет Message Agent, но только в консолидированной базе данных. При обнаружении конфликта операторов UPDATE в сообщении от удаленной базы данных Message Agent инициирует на сервере базы данных две операции:

- 1 Выполняется оператор UPDATE;
- 2 Вызываются любые процедуры разрешения конфликтов.

Операторы UPDATE выполняются даже в том случае, если значения раздела VERIFY не совпадают, независимо от наличия или отсутствия триггера UPDATE RESOLVE.

☞ В консолидированной базе данных Adaptive Server Anywhere применяется другой метод разрешения конфликтов. Для получения дополнительной информации см. раздел "Принципы разрешения конфликтов в SQL Remote" на стр. 103.

Реализация методов разрешения конфликтов

В данном разделе описываются возможности по реализации пользовательских методов разрешения конфликтов в SQL Remote.

Необходимые объекты

Для реализации решения для каждой таблицы, относительно которой необходимо устранить конфликт, следует создать три объекта базы данных.

- ◆ **Таблица старых значений** - для сохранения значений, которые хранились в таблице при получении противоречивого сообщения.
- ◆ **Таблица полученных значений** - для сохранения определенных в сообщении значений, которые сохраняются в таблице в удаленной базе данных при выполнении противоречивого обновления.
- ◆ **Хранимая процедура** - для выполнения действий по разрешению конфликта.

Эти объекты должны существовать только в консолидированной базе данных, поскольку именно в этой базе данных происходит разрешение конфликта. Эти объекты не должны быть включены ни в одну публикацию.

Присваивание имен объектам

Имена объектов разрешения конфликтов задаются с помощью дополнительных параметров хранимой процедуры **sp_add_remote_table** или **sp_modify_remote_table**, с помощью которых таблица выбирается для репликации.

Процедуры **sp_add_remote_table** и **sp_modify_remote_table** принимают один обязательный аргумент, который представляет собой имя таблицы, выбранной для репликации. Этот аргумент принимает три дополнительных аргумента, которые обозначают имена объектов, используемых для разрешения конфликтов. К примеру, синтаксис процедуры **sp_add_remote_table** следующий:

```
exec sp_add_remote_table имя_таблицы
    [ , процедура_разрешения ]
    [ , таблица_старых_значений ]
    [ , таблица_полученных_значений ]
```


Каждый из трех вышеперечисленных объектов (*процедура разрешения*, *таблица старых значений* и *таблица полученных значений*) должен быть создан. Ниже поочередно рассматриваются эти три объекта.

- ◆ **Таблица старых значений.** Эта таблица должна иметь те же имена столбцов и типы данных, что и в таблице, указанной как *имя_таблицы*, и не должна содержать внешних ключей.

При возникновении конфликта строка вставляется в *таблицу старых значений*, содержащую значения строки из таблицы *имя_таблицы* до выполнения обновления этой таблицы. Сразу же после выполнения *процедуры разрешения* эта строка удаляется.

Поскольку Message Agent обрабатывает обновления как набор обновлений отдельных строк, таблица всегда содержит одну единственную строку.

- ◆ **Таблица полученных значений.** Эта таблица должна иметь те же имена столбцов и типы данных, что и в таблице *имя_таблицы*, и не должна содержать внешних ключей.

При возникновении конфликта строка вставляется в *таблицу полученных значений*, содержащую значения строки из таблицы *имя_таблицы* в удаленной базе данных до выполнения обновления этой таблицы. Сразу же после выполнения *процедуры разрешения* эта строка удаляется.

Поскольку Message Agent обрабатывает обновления как набор обновлений отдельных строк, таблица всегда содержит одну единственную строку.

- ◆ **Процедура разрешения.** Эта процедура выполняет любые действия, необходимые для разрешения конфликтов, такие как изменение значения в строке или передача значений в отдельную таблицу.

После создания этих объектов необходимо выполнить процедуру **sp_add_remote_table** или **sp_modify_remote_table**, чтобы отметить эти три объекта как объекты разрешения конфликтов для таблицы.

Ограничения

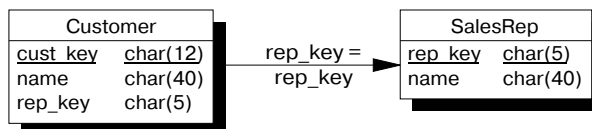
- ◆ Разрешение конфликтов в базе данных Adaptive Server Enterprise не будет выполняться для таблицы, содержащей более 128 столбцов, несмотря на то, что параметр `VERIFY_ALL_COLUMNS` будет иметь значение `ON`. Если даже `VERIFY_ALL_COLUMNS` будет установлен в значение `OFF`, а оператор `UPDATE` будет обновлять более 128 столбцов, разрешение конфликтов производиться не будет.

Пример первого способа разрешения конфликтов

В этом примере информация о конфликтах в таблице Customer из примера с двумя таблицами, представленного в учебных разделах, направляется в отдельную таблицу для последующей обработки.

База данных

База данных с двумя таблицами имеет следующий вид:



Цели использования этого метода разрешения конфликтов

При использовании этого метода разрешения конфликтов сообщение о конфликте обновлений для столбца **name** таблицы Customer будет заноситься в отдельную таблицу под названием **ConflictLog**.

Объекты разрешения конфликтов

Определение таблиц разрешения конфликтов следующее:

```

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 )
)

```

В каждой из этих таблиц имеются точно такие же столбцы и типы данных, что и в таблице Customer. Единственное отличие в их определении заключается в том, что они не имеют внешнего ключа к таблице **SalesRep**.

Процедура разрешения конфликтов передает информацию о конфликтах в таблицу **ConflictLog**, которая имеет следующее определение:

```

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)
)

```

Процедура разрешения конфликтов следующая:

```

CREATE PROCEDURE ResolveCustomer
AS
BEGIN
    DECLARE @cust_key CHAR(12)
    DECLARE @lost_name CHAR(40)
    DECLARE @won_name CHAR(40)
    // Получение имени, которое было
    // удалено из таблицы OldCustomer
    SELECT @lost_name=name,
           @cust_key=cust_key
    FROM OldCustomer

    // Получение вставленного имени
    // из таблицы 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

```

Эта процедура разрешения конфликтов не использует таблицу **RemoteCustomer**.

Принципы разрешения конфликтов

Хранимая процедура является ключевым компонентом при разрешении конфликтов. Она выполняется следующие действия:

1. Получение значения **@lost_name** из таблицы **OldCustomer**, а также значение первичного ключа, чтобы можно было обратиться к реальной таблице. Значением **@lost_name** является значение, переопределенное обновлением, вызвавшим конфликт.
2. Получение значения **@won_name** из таблицы Customer непосредственно. Это значение, которое переопределило значение **@lost_name**. Хранимая процедура выполняется *после* того, как было выполнено обновление; именно поэтому это значение присутствует в таблице Customer. Подобная операция в SQL Remote для Adaptive Server Anywhere выполняется по-другому; там разрешение конфликтов реализуется с помощью триггера BEFORE.

- 3 Добавление строки в таблицу **ConflictLog**, содержащую значения **@lost_name** и **@won_name**.
- 4 После выполнения процедуры Message Agent удаляет эти строки из таблиц **OldCustomer** и **RemoteCustomer**. В этом простом примере строка **RemoteCustomer** не использовалась.

Проверка на примере

❖ Проверка на примере

- 1 Создайте таблицы и процедуру в консолидированной базе данных и добавьте их как объекты разрешения конфликтов в таблицу Customer.
- 2 Внесите и подтвердите изменение в консолидированной базе данных. Например:

```
UPDATE Customer
SET name = 'Sea Sports'
WHERE cust_key='cust1'
go
COMMIT
go
```

- 3 Внесите и подтвердите другое изменение в ту же самую строку в удаленной базе данных. Например:

```
UPDATE Customer
SET name = 'C Sports'
WHERE cust_key='cust1'
go
COMMIT
go
```

- 4 Выполните репликацию изменения из удаленной базы данных в консолидированную базу данных; для этого запустите Message Agent в удаленной базе данных, чтобы переслать сообщение и затем получить и обработать его в консолидированной базе данных.
- 5 В консолидированной базе данных просмотрите таблицы **Customer** и **ConflictLog**. Таблица Customer будет содержать значение из удаленной базы данных:

cust_key	name	rep_key
cust1	C Sports	rep1

В таблице **ConflictLog** будет одна строка с сообщением о конфликте:

conflict_key	lost_name	won_name
1	Sea Sports	C Sports

Пример второго способа разрешения конфликтов

Этот примере иллюстрирует немного более сложный пример разрешения конфликтов; за основу взята та же ситуация, что и в предыдущем примере, который рассматривается в разделе “Пример первого способа разрешения конфликтов” на стр. 169.

Цели использования этого метода разрешения конфликтов

В данном случае разрешение конфликтов преследует следующие цели:

- ◆ Запрет обновления от удаленной базы данных. В предыдущем примере обновление разрешалось.
- ◆ Сообщение имени удаленного пользователя, обновление которого не было выполнено, а также имен для удаленных (lost) и вставленных (won) данных.

Объекты разрешения конфликтов

В данном случае таблица **ConflictLog** имеет дополнительный столбец для записи идентификатора удаленного пользователя. Это следующая таблица:

```
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 )
)
```

Хранимая процедура имеет более сложную структуру. Поскольку обновление скорее будет не выполнено, нежели выполнено, значение **lost_name** теперь будет ссылаться на значение, поступающее в сообщении. Сперва оно сохраняется, но затем процедура разрешения конфликтов заменяет его на прежнее значение.

Хранимая процедура использует данные из временной таблицы **#remote**. Для создания процедуры, которая будет ссылаться на временную таблицу, необходимо сперва создать эту временную таблицу. Для этого используется следующий оператор:

```
CREATE TABLE #remote (
    current_remote_user varchar(128),
    current_publisher varchar(128)
)
```

Эта таблица создается в **TEMPDB** и существует только в текущем сеансе. **Message Agent** создает свою собственную таблицу **#remote** при подключении и использует ее после выполнения процедуры.

```
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)

    -- Получение имени, которое было до обработки
    -- сообщения, из OldCustomer. Данные по этому имени
    -- и будут в итоге вставлены.
    SELECT @won_name=name,
           @cust_key=cust_key
    FROM OldCustomer
    -- Получение имени, которое было сохранено
    -- Message Agent и взято из таблицы Customer.
    -- Данные по этому имени будут удалены
    SELECT @lost_name=name
    FROM Customer
    WHERE cust_key = @cust_key
    -- Получение значения удаленного пользователя из
    -- таблицы #remote
    SELECT @remote_user = current_remote_user
    FROM #remote
    -- Сообщение о конфликте
    INSERT INTO ConflictLog ( lost_name,
                             won_name, remote_user )
    VALUES ( @lost_name, @won_name, @remote_user )

    -- Запрет обновления от Message Agent путем сброса
    -- строки в таблице Customer
    UPDATE Customer
    SET name = @won_name
    WHERE cust_key = @cust_key
END
```

Примечания

Ниже приводится несколько примечаний по теме данного раздела.

- ◆ Идентификатор удаленного пользователя сохраняется в Message Agent в столбце **current_remote_user** временной таблицы **#remote**.
- ◆ Операция UPDATE из Message Agent обрабатывается перед выполнением процедуры, поэтому процедура должна явно заменять значения. В SQL Remote для Adaptive Server Anywhere это происходит по-другому; там разрешение конфликтов выполняется с помощью триггеров BEFORE.

Проверка на примере

❖ Проверка на примере

- 1 Создайте таблицы и процедуру в консолидированной базе данных, а также добавьте их как объекты разрешения конфликтов в таблицу Customer.
- 2 Внесите и подтвердите изменение в консолидированной базе данных. Например:

```
UPDATE Customer
SET name = 'Consolidated Sports'
WHERE cust_key='cust1'
go
COMMIT
go
```

- 3 Внесите и подтвердите другое изменение в ту же самую строку в удаленной базе данных. Например:

```
UPDATE Customer
SET name = 'Field Sports'
WHERE cust_key='cust1'
go
COMMIT
go
```

- 4 Выполните репликацию изменения из удаленной базы данных в консолидированную базу данных; для этого запустите Message Agent в удаленной базе данных, чтобы переслать сообщение и затем получить и обработать его в консолидированной базе данных.
- 5 В консолидированной базе данных просмотрите таблицы **Customer** и **ConflictLog**. Таблица Customer будет содержать значение из удаленной базы данных:

cust_key	name	rep_key
cust1	Consolidated Sports	rep1

В таблице **ConflictLog** будет одна строка, в которой будет сообщаться о конфликте и будет записано значение, введенное в удаленной базе данных:

conflict_key	lost_name	won_name	remote_user
1	Field Sports	Consolidated Sports	field_user

- 6 Снова запустите Message Agent в удаленной базе данных. Из удаленной базы данных будет получено исправленное обновление, и, следовательно, имя клиента будет установлено на значение Consolidated Sports и здесь.

Методы предотвращения ошибок ссылочной целостности

Таблицы в реляционной базе данных связываются посредством ссылок по внешнему ключу. Ограничения ссылочной целостности, применяемые как следствие использования этих ссылок, обеспечивают постоянную непротиворечивость информации в базе данных. При репликации только части

базы данных могут возникнуть проблемы сохранения ссылочной целостности реплицированной базы данных.

При нарушении ссылочной целостности репликация завершается

Если удаленная база данных получает сообщение с оператором, который не может быть выполнен по причине ограничений ссылочной целостности, последующие сообщения в этой базе данных обрабатываться не будут (так как они поступают после сообщения, которое еще не было обработано), включая операторы ретрансляции, которые будут находиться в очереди сообщений.

Этих проблем можно избежать, если уделять должное внимание вопросам ссылочной целостности еще на этапе построения публикации. В данном разделе описываются некоторые наиболее общие проблемы ссылочной целостности, и рассматриваются способы предотвращения их возникновения.

Ошибки при
отсутствии
репликации
связанной таблицей

Рассмотрим следующую публикацию **SalesRepData**:

```
exec sp_add_remote_table 'SalesRep'  
exec sp_create_publication 'SalesRepData'  
exec sp_add_article 'SalesRepData', 'SalesRep'  
go
```

Если бы таблица **SalesRep** имела внешний ключ к другой таблице (предположим, **Employee**), которая не была бы включена в публикацию, репликация вставок или обновлений таблицы **SalesRep** не выполнялась бы до тех пор, пока в удаленной базе данных не были бы удалены ссылки по внешнему ключу.

При использовании утилиты извлечения для создания удаленных баз данных ссылка по внешнему ключу автоматически исключается из удаленной базы данных, и подобной проблемы не возникает. Однако в базе данных не существует ограничений, которые предотвращали бы вставку недопустимых значений в столбец **rep_id** таблицы **SalesRep**; если такое случается, оператор INSERT в консолидированной базе данных выполняться не будет. Для предотвращения возникновения подобной проблемы можно включить в публикацию таблицу **Employee** (или, по крайней мере, ее первичный ключ).

Обеспечение уникальности первичных ключей

Все пользователи на физически удаленных узлах могут включать новые строки в таблицу; следовательно, имеется очевидная проблема обеспечения и поддержания уникальности значений первичных ключей.

Если два пользователя вставляют строку, используя одинаковые значения первичных ключей, оператор INSERT, который поступает в эту базу данных в системе репликации последним из двух, не выполняется. Поскольку SQL Remote является системой репликации для пользователей, выполняющих подключение к базе данных периодически, механизм блокирования для всех баз данных системы репликации может не действовать. Необходимо построить систему SQL Remote таким образом, чтобы ошибки дублирования первичных ключей не возникали.

Для предотвращения ошибок первичных ключей в системе SQL Remote, необходимо обеспечить уникальность первичных ключей таблиц, которые могут быть изменены с нескольких узлов. Для достижения данной цели существует несколько способов. В этой главе рассматривается обобщенный, целесообразный и надежный метод, при котором используется пул значений первичных ключей для каждого узла в системе репликации.

Общие сведения о пулах первичных ключей

Пул первичных ключей представляет собой таблицу, которая содержит набор значений первичных ключей для каждой базы данных в системе SQL Remote. Каждый удаленный пользователь получает свой собственный набор значений первичных ключей. Когда удаленный пользователь вставляет в таблицу новую строку, он использует хранимую процедуру для выбора действительного первичного ключа из пула. Пул поддерживается посредством периодического выполнения в консолидированной базе данных процедуры, которая пополняет запас значений.

Данный способ описывается на примере простой базы данных, в которой хранится информация о торговых представителях и их клиентах. Эти таблицы намного проще по своей структуре тех таблиц, которые использовались бы в реальной базе данных; однако такое упрощение позволяет более подробно рассмотреть аспекты, связанные с репликацией.

Пул первичных ключей

Пул первичных ключей хранится в отдельной таблице. Таблица пула первичных ключей создается с помощью приведенного ниже оператора CREATE TABLE:

```
CREATE TABLE KeyPool (
    table_name VARCHAR(40) NOT NULL,
    value INTEGER NOT NULL,
    location VARCHAR(6) NOT NULL,
    PRIMARY KEY (table_name, value),
)
go
```

Столбцы этой таблицы имеют следующие значения:

Столбец	Описание
table_name	Содержит имена таблиц, для которых должны храниться пулы первичных ключей. В приведенном ниже простом примере, если бы планировалось добавлять новых торговых представителей только в консолидированной базе данных, пул первичных ключей был бы необходим только для таблицы Customer; поэтому данный столбец можно считать

Столбец	Описание
value	избыточным. Он включен в пример для иллюстрации решения в общем виде. Содержит список значений первичных ключей. Каждое значение уникально для каждой таблицы, перечисленной в столбце table_name .
location	В некоторых системах этот идентификатор может совпадать со значением rep_key таблицы SalesRep . В других системах помимо торговых представителей могут быть другие пользователи, поэтому эти два идентификатора должны быть отличными друг от друга.

Для повышения производительности можно создать индекс к таблице:

```
CREATE INDEX KeyPoolLocation
ON KeyPool (table_name, location, value)
go
```

Репликация пула первичных ключей

Можно либо включать пул ключей в существующую публикацию, либо совместно использовать его как отдельную публикацию. В данном примере для пула первичных ключей создается отдельная публикация.

❖ Процедура репликации пула первичных ключей

- 1 Создайте публикацию для данных пула первичных ключей.

```
sp_create_publication 'KeyPoolData'
go
sp_add_remote_table 'KeyPool'
go
sp_add_article 'KeyPoolData', 'KeyPool',
NULL, 'location'
go
```

- 2 Создайте подписки на публикацию **KeyPoolData** для каждой удаленной базы данных.

```
sp_subscription 'create',
KeyPoolData,
field_user,
rep1
go
```

Аргументом подписки является идентификатор местоположения.

В некоторых случаях имеет смысл добавлять таблицу **KeyPool** к существующей публикации и использовать один и тот же аргумент для создания подписок на каждую публикацию. В данном примере местоположение и значения **rep_key** различны, чтобы показать решение в более общем виде.

Заполнение и пополнение пула ключей

Каждый раз, когда пользователь добавляет информацию о новом клиенте, его пул доступных первичных ключей уменьшается на один ключ. Таблица пула первичных ключей в консолидированной базе данных должна периодически пополняться с помощью следующей процедуры:

```
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
            -- убедитесь, что есть 10 значений
            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
```

С помощью данной процедуры пул каждого пользователя пополняется до десяти значений. При работе в реальной ситуации возможно использование большего значения. Данное значение устанавливается исходя из того, как часто пользователи вставляют строки в таблицы в базе данных.

Процедуру **ReplenishPool** необходимо периодически запускать в консолидированной базе данных для пополнения пула значений первичных ключей в таблице **KeyPool**.

Для выполнения процедуры **ReplenishPool** необходимо, чтобы для каждого подписчика существовало по крайней мере одно значение первичного ключа что обеспечит возможность нахождения максимального значения и добавления значения для генерации следующего набора. При первоначальном заполнении пула можно вставить по одному значению для каждого пользователя, после чего вызвать процедуру **ReplenishPool** для заполнения остальных значений. Следующий пример иллюстрирует выполнение данной операции для трех удаленных пользователей и одного консолидированного пользователя по имени **Office**:

```
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
```

Использовать триггер для пополнения пула ключей невозможно

Для пополнения пула ключей нельзя использовать триггер, так как действия, включая действия триггеров, не реплицируются в выполняющую первоначальную операцию удаленную базу данных.

Добавление информации о новых клиентах

Если торговому представителю необходимо добавить в таблицу Customer информацию о новом клиенте, значение первичного ключа, которое требуется вставить, получается с помощью хранимой процедуры. В данном примере рассматривается хранимая процедура для получения значения первичного ключа, а также хранимая процедура для выполнения оператора INSERT.

В данных процедурах используется тот факт, что идентификатором торгового представителя является текущий издатель (CURRENT PUBLISHER) удаленной базы данных.

- ◆ **Процедура NewKey.** Процедура **NewKey** извлекает из пула ключей целочисленное значение, после чего удаляет это значение из пула.

```
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
        -- Не используйте последнее значение,
        -- так как в этом случае ReplenishPool
        -- работать не будет.
        -- Во избежание этого пул ключей должен
        -- оставаться достаточно наполненным.
        SELECT @Value = NULL
END
```

- ◆ **Процедура NewCustomer.** Процедура **NewCustomer** вставляет в таблицу информацию о новом клиенте, используя получаемое процедурой **NewKey** значение для создания первичного ключа.

```
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
```

Данную процедуру можно улучшить за счет проверки значения **@cust**, получаемого из процедуры **NewKey**, на равенство NULL и, таким образом, предотвратить вставки в случаях, когда это значение - NULL.

Проверка пула ключей

❖ Проверка пула первичных ключей

- 1 Заново извлеките удаленную базу данных, используя идентификатор пользователя `field_user`.
- 2 Попробуйте выполнить следующий типовой оператор INSERT на удаленном и консолидированном узлах:

```
EXEC NewCustomer 'Great White North', repl
```

Заключительная информация о пулах первичных ключей

Для применения методов с использованием пула первичных ключей требуются следующие компоненты:

- ◆ **Таблица пула ключей.** Таблица, в которой хранятся действительные значения первичных ключей для каждой базы данных в системе.
- ◆ **Процедура пополнения.** Хранимая процедура, которая осуществляет пополнение таблицы пула ключей.
- ◆ **Совместное использование пула ключей.** Каждая база данных в системе должна иметь подписку на свой собственный набор действительных значений из таблицы пула ключей.
- ◆ **Процедуры ввода данных.** Ввод новых строк производится с помощью хранимой процедуры, которая выбирает из пула следующее действительное значение первичного ключа, после чего удаляет его из пула ключей.

Создание подписок

Чтобы подписаться на публикацию, каждому подписчику должны быть предоставлены полномочия удаленного пользователя; также для данного пользователя должна быть создана подписка. Отдельные элементы подписки различаются в зависимости от того, использует ли публикация выражение подписки или нет.

Подписки без столбца подписки

Чтобы подписать пользователя на публикацию, в которой нет столбца подписки, необходима следующая информация:

- ◆ **Идентификатор пользователя.** Пользователь, который подписывается на публикацию. Данному пользователю необходимо предоставить полномочия доступа REMOTE.
- ◆ **Имя публикации.** Имя публикации, на которую подписывается пользователь.

Следующий оператор создает подписку для пользователя с идентификатором **SamS** на публикацию **pub_orders_samuel_singer**, которая была создана с использованием столбца подписки:

```
sp_subscription 'create',
               'pub_orders_samuel_singer',
               'SamS'
```

Подписки со столбцом подписки

Чтобы подписать пользователя на публикацию, в которой есть столбец подписки, необходима следующая информация:

- ◆ **Идентификатор пользователя.** Пользователь, который подписывается на публикацию. Данному пользователю необходимо предоставить полномочия доступа REMOTE.
- ◆ **Имя публикации.** Имя публикации, на которую подписывается пользователь.
- ◆ **Значение подписки.** Значение, которое должно проверяться выражением подписки публикации. Например, если в публикации есть имя столбца, в котором хранится идентификатор служащего в виде выражения подписки, значение идентификатора подписывающегося пользователя должно быть указано в подписке. Значение подписки всегда является строкой.

Следующий оператор создает подписку для Сэмюэля Сингера (Samuel Singer, идентификатор пользователя **SamS**, идентификатор служащего 856) на публикацию **pub_orders**, определенную выражением подписки **sales_rep**, которая запрашивает строки с информацией о собственных продажах Сэмюэля Сингера:

```
sp_subscription create,
               pub_orders,
               SamS,
               '856'
```

Активизация подписки

Чтобы правильно получать и выполнять обновления, каждый подписчик должен иметь первоначальную копию данных. Процесс синхронизации рассматривается в разделе "Синхронизация баз данных" на стр. 163.

Администрирование SQL Remote

В этой части рассматриваются вопросы развертывания и администрирования системы SQL Remote.



Развертывание и синхронизация баз данных

Об этой главе

В данной главе описываются действия, которые необходимо предпринять для развертывания и синхронизации инсталляции репликации SQL Remote.

Содержание

Раздел	Страница
Общие сведения о развертывании системы	160
Тестирование перед развертыванием системы	161
Синхронизация баз данных	163
Использование утилиты извлечения	165
Синхронизация данных по системе передачи сообщений	172

Общие сведения о развертывании системы

Задачи развертывания	<p>По завершении этапа настройки SQL Remote следующим шагом следует осуществить создание и развертывание удаленных баз данных и приложений.</p>
	<p>В некоторых случаях развертывание системы является наиболее важной и трудоемкой задачей. Например, при наличии в системе автоматизации торговой деятельности большого количества удаленных пользователей процесс развертывания системы будет включать в себя следующие этапы:</p>
	<ol style="list-style-type: none">1 Создание базы данных Adaptive Server Anywhere для каждого удаленного пользователя со своей собственной копией использованием начальных копий данных, имеющихся у этих пользователей;2 Установка базы данных, а также ПО сервера базы данных Adaptive Server Anywhere, SQL Remote Message Agent и клиентских приложений, на машине каждого пользователя.3 Создание требуемой конфигурации системы, включая назначение правильных имен пользователей, строк подключения Message Agent, полномочий и т.д.
	<p>При развертывании крупномасштабной системы на удаленных узлах, как правило, устанавливаются базы данных Adaptive Server Anywhere; в данной главе более подробно рассматриваются именно такие случаи.</p>
Вопросы, рассматриваемые в данной главе	<p>В данной главе рассматриваются следующие вопросы:</p> <ul style="list-style-type: none">◆ Создание удаленных баз данных. Перед развертыванием системы SQL Remote на каждом удаленном узле необходимо создать удаленную базу данных. В данном разделе описывается в основном создание удаленных баз данных Adaptive Server Anywhere.◆ Синхронизация данных. Синхронизация базы данных – это установка первоначальной копии данных в удаленной базе данных.

Тестирование перед развертыванием системы

Перед развертыванием системы SQL Remote, особенно при наличии большого количества удаленных узлов, необходимо выполнить полное тестирование системы.

На этапе проектирования и настройки системы можно внести изменения во многие аспекты функционирования SQL Remote. Изменение публикаций, типов сообщений, написание триггеров для разрешения конфликтов обновлений – все это выполняется с помощью простого набора действий.

По завершении развертывания системы SQL Remote ситуация меняется. Настройки и установки системы SQL Remote можно рассматривать как одну **рассредоточенную базу данных**, которая находится на большом количестве узлов и в которой согласованность данных поддерживается в свободном режиме. Состояние данных во всех базах данных, входящих в состав системы, постоянно меняется, однако все изменения в данных периодически реплицируются по всей системе в виде законченных транзакций. Согласованность данных в системе SQL Remote достигается за счет тщательного проектирования публикаций, а также посредством разрешения конфликтов обновлений (UPDATE) по мере их возникновения.

Обновление и ресинхронизация

После развертывания и запуска системы SQL Remote вносить изменения в конфигурацию будет практически невозможно. Установку новых версий SQL Remote необходимо производить с такой же осторожностью, как и при первоначальном развертывании системы. Это в равной степени относится к установке более поздних или исправленных версий программного обеспечения баз данных Adaptive Server Enterprise или Adaptive Server Anywhere. При любом подобном обновлении перед развертыванием программное обеспечение должно быть протестировано на совместимость.

Внесение изменений в схему базы данных в одной базе данных внутри системы может привести к сбоям в работе из-за несовместимости объектов базы данных. Режим ретрансляции позволяет рассылать изменения схемы в некоторые или во все базы данных системы SQL Remote, однако использовать его нужно с большой осторожностью и после тщательного планирования операции.

В рассредоточенной базе данных при свободном режиме согласования происходит постоянное обновление данных. При этом остановить процесс внесения изменений во все базы данных для обновления схемы базы данных и последующего перезапуска системы невозможно.

При отсутствии должного планирования изменения в схеме базы данных приведут к возникновению ошибок по всей системе, после чего потребуется деактивизировать все подписки и выполнить их ресинхронизацию. Ресинхронизация включает в себя загрузку новых копий данных в каждую удаленную базу данных, что для многих пользователей является трудоемким процессом, который приводит к прерыванию работы и возможной потере данных.

Нежелательные изменения в запущенной системе

Ниже приведены примеры изменений, которые не следует вносить в развернутую и уже запущенную систему SQL Remote. Из списка можно увидеть, что некоторые изменения можно назвать нежелательными, но **разрешенными**, т.е. в целом допустимыми, тогда как другие изменения являются **запрещенными** и безоговорочно недопустимыми.

Следующие действия выполнять не следует, за исключением особо оговоренных случаев:

- ◆ Изменение издателя для консолидированной базы данных.

- ◆ Внесение запрещенных изменений в таблицы, например, удаление столбца или запрещение использования в столбце значения NULL. Сообщения с изменениями в записях удаляемого столбца или записях со значением NULL могут уже быть переданы в базы данных по всей системе, поэтому такое обновление приведет к невозможности реализации этих изменений.
- ◆ Изменение публикации. Определения публикации должны сохраняться как на локальном, так и на удаленном узлах, поскольку изменения, построенные на основе старого определения публикации, могут уже быть переданы в базы данных по всей системе SQL Remote.

Допускается вносить только разрешенные изменения, такие как добавление новой таблицы или столбца, при условии применения ретрансляции, что исключит возможность возникновения ситуации отсутствия в удаленной базе данных и в публикации на удаленной базе данных новой таблицы или столбца.

- ◆ Удаление подписки. Удалять подписку можно только в том случае, если для удаления данных из удаленного узла используются операторы delete в режиме ретрансляции.
- ◆ Выгрузка и перезагрузка базы данных Adaptive Server Anywhere.

Если база данных Adaptive Server Anywhere участвует в репликации, производить ее выгрузку и перезагрузку без ресинхронизации базы данных нельзя. Репликация выполняется на основе данных журнала транзакций, а после выгрузки и перезагрузки базы данных старый журнал транзакций станет недоступным. Поэтому при участии базы данных в процессе репликации особую важность имеет выполнение периодического резервного копирования.

Выгрузка и перезагрузка базы данных Adaptive Server Enterprise может производиться только после остановки процесса репликации и завершения сканирования журнала транзакций. При этом необходимо переустановить строки **page_id** и **row_id** в таблице **sr_queue_state** очереди с сохранением.

Синхронизация баз данных

Определение синхронизации	Репликация SQL Remote выполняется на основе информации журнала транзакций. Однако возможны два случая, когда SQL Remote удаляет все существующие строки из тех таблиц удаленной базы данных, которые составляют часть публикации, и копирует полное содержание публикации из консолидированной базы данных в удаленный узел. Этот процесс называется синхронизацией .
Случаи применения синхронизации	Синхронизация применяется при следующих обстоятельствах: <ul style="list-style-type: none"> ◆ Синхронизация выполняется при создании подписки в консолидированной базе данных, что обеспечивает запуск удаленной базы данных с тем же состоянием данных, что и в консолидированной базе данных. ◆ Функция синхронизации позволяет возобновить правильный обмен информацией между удаленной и консолидированной базами данных при повреждении удаленной базы данных, неправильном реагировании на запросы консолидированной базы данных и невозможности ее восстановления посредством ретрансляции SQL.
Способы синхронизации	Синхронизация удаленной базы данных может быть выполнена следующими способами: <ul style="list-style-type: none"> ◆ Синхронизация с помощью утилиты извлечения базы данных. Эта утилита создает схему удаленной базы данных Adaptive Server Anywhere и синхронизирует удаленную базу данных. В большинстве случаев рекомендуется использовать именно этот способ. ◆ Синхронизация вручную. Синхронизация удаленной базы данных вручную производится путем загрузки из файлов с использованием конвейера PowerBuilder или любого другого инструментального средства. ◆ Синхронизация по системе передачи сообщений. Синхронизация удаленной базы данных через систему передачи сообщений с использованием оператора SYNCHRONIZE SUBSCRIPTION (Adaptive Server Anywhere) или процедуры <code>sp_subscription 'synchronize'</code> (Adaptive Server Enterprise).

Предостережение

Не выполняйте SYNCHRONIZE SUBSCRIPTION и sp_subscription 'synchronize' в удаленной базе данных.

Извлечение базы данных при различных операционных системах

Часто система строится так, что консолидированный сервер и удаленные базы данных работают на различных операционных системах.

Базы данных Adaptive Server Anywhere можно скопировать из одного файла в другой или из одной операционной системы в другую. Это дает определенную гибкость в выборе способа первоначальной синхронизации баз данных.

Пример

К примеру, на сервере Adaptive Server Enterprise с консолидированной базой данных установлена система UNIX, а удаленные базы данных требуется разместить на портативных компьютерах с одной из разновидностей ОС Windows.

В этом случае при наличии необходимого программного обеспечения возможно несколько вариантов выбора платформы для извлечения базы данных. Ниже приведены некоторые из них:

- ◆ Запуск утилиты извлечения в системе UNIX для создания сценария перезагрузки и файлов данных. Создание копии сценария и файлов данных

на машине с ОС Windows. Создание баз данных Adaptive Server Anywhere и их загрузка (вместе со схемой и данными) в Windows.

- ◆ Запуск утилиты извлечения в системе UNIX для создания сценария перезагрузки и файлов данных. Создание баз данных Adaptive Server Anywhere и их загрузка (вместе со схемой и данными) в ту же систему UNIX, после чего создание копии файлов баз данных на машине с ОС Windows для развертывания.
- ◆ Запуск утилиты извлечения в ОС Windows и выполнение всех необходимых действий по созданию базы данных в ОС Windows.

Примечания относительно синхронизации и извлечения

- ◆ Извлечение большого количества подписок или синхронизация подписок с большими, часто используемыми таблицами может замедлить доступ к базе данных для других пользователей. Проводить извлечение таких подписок следует тогда, когда база данных используется неинтенсивно. Это также может быть выполнено автоматически при использовании раздела SEND AT с указанием наиболее подходящего времени.
- ◆ Синхронизация выполняется для всей подписки. В настоящее время не существует простого способа синхронизации отдельной таблицы.

☞ Для получения дополнительной информации по способам повышения производительности систем Adaptive Server Enterprise со столбцом списка подписки см. разделы "Повышение производительности при операции извлечения" на стр. 156 и "Повышение производительности при операции извлечения совместно используемых строк" на стр. 163.

Использование утилиты извлечения

Утилита извлечения является одним из средств создания удаленных баз данных Adaptive Server Anywhere. Использовать эту утилиту для создания удаленных баз данных Adaptive Server Enterprise нельзя.

Запуск утилиты извлечения

Вызвать утилиту извлечения можно следующими способами:

- ◆ Из Sybase Central, если в качестве консолидированной базы данных используется Adaptive Server Anywhere.
- ◆ Из командной строки. Это утилита *dbxtract* (в Adaptive Server Anywhere) или *ssxtract* (в Adaptive Server Enterprise).

Предостережение

Не запускайте Message Agent при запущенной утилите извлечения. Результаты в этом случае непредсказуемы.

Создание базы данных из файлов перезагрузки

Утилита командной строки выгружает схему базы данных и данные, необходимые для создания удаленной базы данных Adaptive Server Anywhere для указанного подписчика. Она создает командный файл SQL с именем по умолчанию *reload.sql* и набор файлов данных. Эти файлы можно использовать для создания удаленной базы данных Adaptive Server Anywhere.

Необходимость редактирования reload.sql

Утилита извлечения базы данных предназначена для того, чтобы упростить процесс подготовки удаленных баз данных, однако она не является универсальным средством, подходящим для всех возможных ситуаций. При создании удаленных баз данных иногда требуется отредактировать командный файл *reload.sql*.

❖ Процедура создания удаленной базы данных из файла перезагрузки

- 1 Создайте базу данных Adaptive Server Anywhere с помощью одного из следующих средств:
 - ◆ мастер создания базы данных Sybase Central (Create Database), расположенный в папке Utilities;
 - ◆ утилита *dbinit*.
- 2 Выполните подключение к базе данных из утилиты Interactive SQL и запустите командный файл *reload.sql*. Приведенный ниже оператор, вводимый в окне SQL Statements, запускает командный файл *reload.sql*:

```
read путь \reload.sql
```

где *путь* - это путь к командному файлу перезагрузки.

При использовании Sybase Central утилита извлечения производит выгрузку базы данных тем же самым способом, что и *dbxtract*, после чего выполняет дополнительные операции по созданию новой базы данных.

Утилита извлечения не использует систему передачи сообщений. Файл перезагрузки (*ssxtract/dbxtract*) или база данных (из Sybase Central) создаются в папке, к которой имеется доступ из данной машины. Синхронизация большого количества подписок в системе передачи сообщений может привести к большому объему трафика сообщений; если система передачи сообщений недостаточно

надежна, может потребоваться некоторое время на то, чтобы все сообщения были правильно получены в удаленных узлах.

Подготовка к извлечению базы данных

Перед использованием утилиты извлечения в консолидированной базе данных необходимо выполнить следующие шаги.

- ◆ Создание типа сообщений для использования при репликации;
- ◆ Добавление в базу данных идентификатора пользователя-издателя;
- ◆ Добавление в базу данных удаленных пользователей;
- ◆ Добавление в базу данных публикации;
- ◆ Создание подписки для удаленных пользователей;
- ◆ Если требуется указать параметры системы передачи сообщений, необходимо их установить.

☞ Для получения информации о процедурах выполнения этих шагов см. учебный раздел в главе "Учебные разделы для пользователей Adaptive Server Anywhere" на стр. 25. Для получения информации о процедурах установки параметров системы передачи сообщений см. раздел "Установка параметров управления типами сообщений" на стр. 186.

При использовании утилиты извлечения для создания удаленной базы данных, пользователь, для которого создается база данных, должен иметь такие же полномочия доступа, какие он имеет в консолидированной базе данных. Более того, если пользователь является членом каких-либо групп в консолидированной базе данных, идентификаторы этих групп создаются в удаленной базе данных с полномочиями, имеющимися в консолидированной базе данных.

Использование утилиты извлечения из Sybase Central

В данном разделе описываются способы извлечения базы данных из текущей консолидированной базы данных для удаленного пользователя. Материал данного раздела относится только к консолидированным базам данных Adaptive Server Anywhere.

После выполнения шагов, указанных мастером извлечения, на пользовательской машине будут произведены следующие операции:

- ◆ Создание удаленной базы данных;
- ◆ Извлечение (выгрузка) в файлы соответствующих структур и/или данных из консолидированной базы данных;
- ◆ Загрузка этих файлов в новую созданную удаленную базу данных

❖ Процедура извлечения базы данных для удаленного пользователя

- 1 Откройте папку Utilities (эта папка находится в папке сервера).
- 2 В правой области окна дважды щелкните по пункту Extract Database.
- 3 Выполняйте указания мастера.

Примечания

- ◆ Обратиться к мастеру можно также последовательным выбором Tools►Adaptive Server Anywhere►Extract Database.

- ◆ При использовании мастера для извлечения незапущенной базы данных мастер может выполнить только выгрузку структуры и данных, но не создание и перезагрузку удаленной базы данных. Поэтому рекомендуется всегда выполнять извлечение из консолидированной базы данных, к которой существует подключение, из Sybase Central.
- ◆ Запускать мастера извлечения можно также для отдельной базы данных или отдельного удаленного пользователя: Sybase Central автоматически установит соответствующие параметры в мастере.
- ◆ Мастер извлечения всегда извлекает (синхронизирует) удаленную базу данных с помощью параметра WITH SYNCHRONIZATION. В тех редких случаях, когда данный параметр не требуется, вместо мастера извлечения следует использовать утилиту *dbxtract*.

Дополнительная информация

Для получения информации о параметрах утилиты извлечения, которые задаются либо из командной строки, либо в мастере извлечения, см. раздел "Параметры утилиты извлечения" на стр. 267.

Построение эффективной процедуры извлечения

При создании большого числа удаленных баз данных путем последовательного запуска утилиты извлечения для каждой базы данных потребуется слишком много времени и ресурсов. Существует возможность выполнить все необходимые действия более удобным и результативным способом. В данном разделе описывается способ повышения эффективности при создании удаленных баз данных с помощью утилиты извлечения.

☞ Для получения дополнительной информации по способам повышения производительности систем Adaptive Server Enterprise со столбцом списка подписки см. разделы "Повышение производительности при операции извлечения" на стр. 156 и "Повышение производительности при операции извлечения совместно используемых строк" на стр. 163.

Существует несколько возможных причин неэффективности процесса извлечения для большого количества баз данных:

- ◆ Утилита извлечения извлекает одну базу данных за раз, включая схему и данные для каждого пользователя. Однако обычно для множества пользователей применяется одна и та же схема, а различаются при этом только данные. При применении самого простого способа извлечения, а именно последовательного запуска утилиты извлечения для каждого отдельного пользователя, происходит ненужное повторение большого количества действий. Для решения этой проблемы извлечение схемы и данных нужно проводить раздельно.
- ◆ При запуске из Sybase Central утилита извлечения создает новую базу данных для каждого пользователя. Если подписчики используют одну и ту же схему, можно было бы создать одну базу данных со схемой, но без данных, и затем копировать этот файл различным пользователям.
- ◆ По умолчанию утилита извлечения выполняется на нулевом уровне изоляции. При извлечении базы данных с активного сервера утилиту необходимо запускать на третьем уровне изоляции (см. раздел "Параметры утилиты извлечения" на стр. 267) для обеспечения согласованности данных в извлеченной базе данных и на сервере.

При запуске утилиты на третьем уровне изоляции возможно увеличение времени обратной передачи на сервере от других пользователей из-за большого количества требуемых блокировок. Рекомендуется запускать утилиту извлечения при низкой нагрузке на сервер, либо запускать ее из копии базы данных.

Эффективный метод извлечения большого количества баз данных

Ниже описывается способ, при котором удастся избежать возникновения приведенных выше проблем.

- 1 Создайте копию консолидированной базы данных и в это же самое время активизируйте подписки из текущей базы данных. После этого начнется рассылка сообщений подписчикам, которая будет осуществляться даже в случае отсутствия соответствующих баз данных и невозможности получения этих сообщений.

Для активизации нескольких подписок в пределах одной транзакции используется оператор REMOTE RESET (Adaptive Server Anywhere) или процедура `sp_remote` (Adaptive Server Enterprise).

- 2 Выполните извлечение удаленных баз данных из консолидированной копии базы данных. Поскольку эта база данных является копией, блокировок и проблем параллельной обработки не возникает. При извлечении большого количества удаленных баз данных этот процесс может занять несколько дней.
- 3 После создания удаленной базы данных данные в ней не являются актуальными, но ее пользователь может получать и обрабатывать сообщения, отсылаемые текущей консолидированной базой данных, на основе которых восстанавливается актуальность данных в удаленной базе данных.

При применении этого метода вмешиваться в работу реальной базы данных приходится только на первом этапе. Если при использовании базы данных устанавливается большое количество блокировок, копию необходимо создавать на третьем уровне изоляции. Кроме того, одновременно с началом создания копии необходимо провести активизацию подписок. Результаты любых операций, производимых в период между созданием копии и активизацией подписок, будут потеряны. Кроме того, такие операции также могут привести к возникновению ошибок в удаленных базах данных.

Извлечение групп

Если удаленный пользователь имеет идентификатор пользователя группы, утилита извлечения извлекает все идентификаторы членов этой группы. Эта функция может применяться по отношению ко всем пользователям в каждой удаленной базе данных с использованием различных идентификаторов пользователей, что позволяет избежать необходимости реализации особых методов извлечения.

При извлечении базы данных для одного пользователя извлекаются все параметры системы передачи сообщений для этого пользователя, а также для групп, членом которых данный пользователь является.

Ограничения при использовании утилиты извлечения

Несмотря на то, что утилита извлечения является рекомендованным средством создания и синхронизации удаленных баз данных из консолидированных баз данных, существуют некоторые обстоятельства, при которых использовать эту утилиту оказывается невозможно. В этих случаях необходимо производить синхронизацию удаленных баз данных вручную. В этом разделе описываются несколько таких ситуаций.

- ◆ **Невозможно создать удаленные базы данных Adaptive Server Enterprise.** Утилита извлечения может быть использована только для создания удаленных баз данных Adaptive Server Anywhere.

- ◆ **Дополнительные таблицы в удаленной базе данных.** Удаленные базы данных могут иметь таблицы, которые отсутствуют в консолидированной базе данных, если эти таблицы не участвуют в репликации. Очевидно, утилита извлечения не может извлечь такие таблицы из консолидированной базы данных.
- ◆ **Различия между Adaptive Server Enterprise и Adaptive Server Anywhere.** Некоторые средства, входящие в состав Adaptive Server Enterprise, отсутствуют в Adaptive Server Anywhere. Утилита извлечения выполняет отображение на схожие средства, но это отображение не является полным.

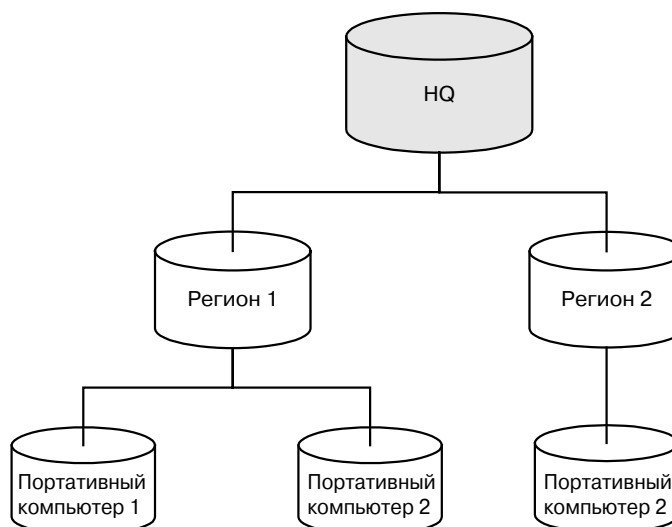
☞ Для получения дополнительной информации по вопросам использования утилиты с Adaptive Server Enterprise и Adaptive Server Anywhere см. раздел "Использование утилиты извлечения для Adaptive Server Enterprise" на стр. 170.

- ◆ **Извлечения процедур и представлений.** По умолчанию утилита извлечения извлекает из базы данных все хранимые процедуры и представления. В то время как некоторые из этих представлений и процедур, вероятно, действительно необходимы на удаленном узле, другие могут оказаться невостребованными - они могут ссылаться только на те части базы данных, которые не используются на удаленном узле.

После выполнения утилиты извлечения необходимо отредактировать сценарий перезагрузки и удалить ненужные представления и процедуры.

- ◆ **Использование утилиты извлечения в многоуровневых системах.** Для понимания того, какую роль утилита извлечения играет в многоуровневых системах, рассмотрим трехуровневую систему SQL Remote.

Такая система иллюстрируется на следующей диаграмме.



Для создания баз данных второго уровня утилиту извлечения можно использовать из консолидированной базы данных на верхнем уровне. После этого к этим базам данных второго уровня можно добавлять удаленных пользователей и использовать утилиту извлечения из каждой базы данных второго уровня для создания удаленных баз данных. Однако если необходимо из консолидированной базы данных верхнего уровня повторно извлечь базы данных второго уровня, созданные удаленные пользователи, а также их подписки и полномочия, будут удалены, и этих пользователей придется восстанавливать. Исключением является случай, когда синхронизируются только данные: тогда утилиту извлечения можно использовать для замены данных в базе данных без изменения схемы.

Использование утилиты извлечения для Adaptive Server Enterprise

Утилита извлечения для Adaptive Server Enterprise берет схему базы данных Adaptive Server Enterprise и создает базу данных Adaptive Server Anywhere. При использовании этого инструментального средства необходимо учитывать ряд особых ограничений и приемов.

Функции Adaptive Server Enterprise, не поддерживаемые в Adaptive Server Anywhere

В Adaptive Server Enterprise есть несколько функций, которые либо не поддерживаются, либо лишь частично поддерживаются в Adaptive Server Anywhere. Некоторые из таких средств утилита извлечения обрабатывает частично, другие не обрабатывает совсем.

☞ Для получения полной информации по совместимости Adaptive Server Enterprise и Adaptive Server Anywhere см. часть "*Совместимость с Transact-SQL*" (*Transact-SQL Compatibility*) в документе "*Руководство пользователя Adaptive Server Anywhere*" (*Adaptive Server Anywhere User's Guide*).

Следующие функции не поддерживаются в *ssxtract*:

- ◆ **Сгруппированные процедуры.** Adaptive Server Anywhere не поддерживает группирование процедур, и группы процедур *ssxtract* извлечь не может.
- ◆ **Поименованные ограничения и значения по умолчанию.** Adaptive Server Anywhere не поддерживает поименованные ограничения и поименованные значения по умолчанию. Любые такие объекты извлекаются непосредственно как ограничения и значения по умолчанию, которые применяются к отдельному объекту; назначенное имя при этом не сохраняется.
- ◆ **Роли.** Утилита *ssxtract* извлекает роли, используя концепцию групп Adaptive Server Anywhere. Он создает группу с поименованной ролью и назначает в нее пользователей.
- ◆ **Пароли.** Если пользователь, для которого извлекается база данных, не имеет учетной записи в SYSLOGINS, пароль не извлекается. Если у этого пользователя имеется имя пользователя, то извлекается фиктивный пароль.
- ◆ **NCHAR, NVARCHAR.** Эти типы данных извлекаются как CHAR и VARCHAR, при этом допускаются значения NULL.
- ◆ **Столбцы меток времени.** Несмотря на то, что в Adaptive Server Anywhere столбец метки времени действительно поддерживается, этот тип данных отличается от аналогичного в Adaptive Server Enterprise. Поэтому столбцы меток времени не извлекаются.

Настройка системных таблиц

Объекты, которые должны быть загружены в базу данных Adaptive Server Anywhere, описываются в системном каталоге. Утилита извлечения для Adaptive Server Enterprise сперва создает в TEMPDB набор системных таблиц Adaptive Server Anywhere, после чего заполняет их данными из каталога Adaptive Server Enterprise. Затем она выгружает этот набор таблиц для создания сценария перезагрузки, который в свою очередь формирует базу данных Adaptive Server Anywhere.

Возможны ситуации, когда необходимо изменить содержание системных таблиц Adaptive Server Anywhere, хранящихся в TEMPDB. В SQL Remote имеются средства для выполнения этой операции.

Хранимой процедурой, которая создает и заполняет системные объекты Adaptive Server Anywhere в TEMPDB, является процедура **sp_populate_sql_anywhere**. По завершении своей работы эта процедура вызывает процедуру под именем

sp_user_extraction_hook. По умолчанию эта процедура не выполняет никаких операций. При необходимости настроить процедуру извлечения это можно сделать путем написания соответствующей процедуры **sp_user_extraction_hook**.

Синхронизация данных по системе передачи сообщений

Создание подписок Подписка создается в консолидированной базе данных Adaptive Server Enterprise с помощью процедуры **sp_subscription** с первым аргументом **create**.

При создании подписки определяются данные, которые должны быть получены. При этом не производится ни синхронизация подписки (предоставление начальной копии данных), ни ее активизация (обмен сообщениями).

Синхронизация подписок

При синхронизации подписки Message Agent посылает подписчику копию всех строк в подписке. При этом предполагается, что соответствующая схема базы данных не изменялась. В консолидированной базе данных Adaptive Server Anywhere подписки синхронизируются с помощью оператора SYNCHRONIZE SUBSCRIPTION. В консолидированной базе данных Adaptive Server Enterprise синхронизация подписок осуществляется с помощью процедуры **sp_subscription** с первым аргументом **synchronize**.

При поступлении в базу данных подписчика сообщений синхронизации Message Agent заменяет текущее содержание базы данных новой копией. Любые данные подписчика, которые являются частью подписки и которые не реплицировались в консолидированную базу данных, удаляются. После завершения процесса синхронизации Message Agent активизирует подписку с помощью оператора START SUBSCRIPTION или процедуры **sp_subscription** с первым аргументом **start**.

Последствия передачи большого объема сообщений

Синхронизация баз данных по системе передачи сообщений может привести к передаче больших объемов сообщений. Во многих случаях предпочтительно использовать процесс извлечения, чтобы синхронизировать базу данных локально, без создания излишней нагрузки на систему передачи сообщений.

Синхронизация подписок во время работы

Если удаленная база данных при обмене информацией с консолидированной базой данных вызывает задержку в передаче, которая не может быть восстановлена применением средств ретрансляции SQL Remote, это можно сделать путем синхронизации подписки, при которой происходит копирование строк подписки из консолидированной базы данных поверх данных подписки в удаленной базе данных.

Потеря данных при синхронизации

Любые данные в удаленной базе данных, которые являются частью подписки, но которые не были реплицированы в консолидированную базу данных, при синхронизации подписки удаляются. При необходимости перед синхронизацией удаленной базы данных можно выполнить ее выгрузку или резервное копирование с помощью Sybase Central или, для Adaptive Server Anywhere, утилиты *dbunload*.

Администрирование SQL Remote

Об этой главе

В данной главе описаны основные положения и принципы администрирования запущенной инсталляции SQL Remote.

☞ Для получения информации по отдельным системам см. главы "Администрирование SQL Remote для Adaptive Server Enterprise" на стр. 229 и "Администрирование SQL Remote для Adaptive Server Anywhere" на стр. 209.

Содержание

Раздел	Страница
Обзор принципов управления	174
Управление полномочиями SQL Remote	175
Управление типами сообщений	183
Работа с Message Agent	193
Повышение производительности Message Agent	197
Кодирование и сжатие сообщений	203
Система отслеживания сообщений	205

Обзор принципов управления

В данной главе описаны принципы администрирования систем SQL Remote.

Администрирование развернутой и запущенной системы SQL Remote осуществляется в консолидированной базе данных.

- ◆ **Полномочия.** Поскольку система SQL Remote включает в себя много различных физических баз данных, для пользователей, имеющих полномочия для работы как с удаленными, так и консолидированными базами данных, необходима схема согласованного назначения этих полномочий. В одном из разделов данной главы описаны положения, которые необходимо учитывать при назначении пользователям полномочий.
- ◆ **Конфигурирование систем передачи сообщений.** Необходимо задать параметры управления и другие настройки для каждой системы передачи сообщений, используемой при репликации SQL Remote. Эти параметры рассматриваются в данной главе.
- ◆ **Message Agent.** Message Agent служит для отправки и получения сообщений. Некоторые особенности работы Message Agent и его конфигурационных параметров различаются для Adaptive Server Anywhere и Adaptive Server Enterprise, однако при этом существуют принципы и методы, общие для обеих систем. Эти общие особенности и рассматриваются в данной главе.
- ◆ **Отслеживание сообщений.** Под администрированием систем SQL Remote подразумевается управление большим количеством сообщений, передаваемых между большим количеством баз данных. Раздел о системе отслеживания сообщений SQL Remote включен с целью оказания пользователю помощи в понимании содержания сообщений, времени их отправки, способов обработки и т.д.
- ◆ **Управление журналом.** SQL Remote получает данные для отправки из журнала транзакций. Поэтому надлежащее управление журналом транзакций и необходимые процедуры создания резервных копий очень важны для бесперебойной работы системы SQL Remote. Несмотря на то, что ответы на многие вопросы зависят от конкретной серверной конфигурации, в данной главе рассматриваются общие положения.
- ◆ **Режим ретрансляции.** Режим ретрансляции обеспечивает возможность непосредственного доступа к удаленному узлу из консолидированной базы данных. Этот способ рассматривается в данной главе.

Управление полномочиями SQL Remote

Пользователи базы данных, задействованной в репликации SQL Remote, идентифицируются по одному из следующих наборов полномочий:

- ◆ **PUBLISH.** Единственный идентификатор пользователя в базе данных идентифицируется для этой базы данных как издатель. Все исходящие сообщения SQL Remote, включая обновления публикаций и подтверждения получения, идентифицируются по идентификатору издателя. Каждая база данных в системе SQL Remote должна иметь один идентификатор издателя, поскольку передача сообщений осуществляется из каждой базы данных в системе SQL Remote.
- ◆ **REMOTE.** Всем получателям сообщений из текущей базы данных или отправителям сообщений в текущую базу данных, находящимся в иерархии SQL Remote непосредственно ниже текущей базы данных, необходимо предоставить полномочия REMOTE.
- ◆ **CONSOLIDATE.** Только одному пользователю в базе данных можно предоставить полномочия CONSOLIDATE. Полномочия CONSOLIDATE идентифицируют базу данных, находящуюся непосредственно над текущей базой данных в системе SQL Remote. Каждая база данных может иметь только одну консолидированную базу данных непосредственно над собой в иерархии.

Информация об этих полномочиях содержится в системных таблицах SQL Remote и не зависит от установки полномочий других баз данных.

Предоставление и отмена полномочий PUBLISH

При отправке базой данных сообщения в нем содержится идентификатор пользователя, представляющий эту базу данных, с целью идентификации источника сообщения получателем. Этот идентификатор пользователя является идентификатором **издателя** базы данных. У одной базы данных может быть только один издатель. В Sybase Central при открытии папки SQL Remote можно в любой момент получить информацию об издателе базы данных Adaptive Server Anywhere.

Издатель требуется даже для удаленных баз данных, доступных в системе репликации только для чтения, поскольку даже эти базы данных отправляют подтверждения в консолидированную базу данных, обеспечивая, таким образом, наличие информации о состоянии репликации. Оператор GRANT PUBLISH для удаленных баз данных Adaptive Server Anywhere выполняется автоматически утилитой извлечения базы данных.

Предоставление и отмена полномочий PUBLISH из Sybase Central

Полномочия PUBLISH для базы данных Adaptive Server Anywhere можно предоставить из Sybase Central. Необходимо выполнить подключение к базе данных как пользователь с полными полномочиями системного администратора или администратора базы данных.

❖ Процедура создания нового пользователя как издателя (Sybase Central)

- 1 Откройте папку Users & Groups.
- 2 Дважды щелкните по пункту Add User.
- 3 В первом окне мастера введите имя и нажмите кнопку Next.
- 4 В следующем окне введите пароль и нажмите кнопку Next.
- 5 В следующем окне проверьте, что пользователю предоставлены полномочия администратора удаленной БД (Remote DBA); это позволяет данному

пользователю запускать Message Agent. Для завершения создания пользователя нажмите кнопку Finish.

❖ **Процедура назначения существующему пользователю статуса издателя (Sybase Central)**

- 1 Выполните одно из следующих действий:
 - ◆ В папке Users & Groups щелкните правой кнопкой по пользователю и выберите во всплывающем меню пункт Change to Publisher.
 - ◆ В папке SQL Remote дважды щелкните по пункту Set Publisher. В открывшемся диалоге выберите пользователя и нажмите OK для задания статуса этого пользователя как издателя базы данных.

Из Sybase Central также можно отменить предоставление полномочий PUBLISH.

❖ **Процедура отмены полномочий PUBLISH (Sybase Central)**

- 1 Выполните одно из следующих действий:
 - ◆ В папке Users & Groups щелкните правой кнопкой по пользователю, которому предоставлены полномочия PUBLISH, и выберите во всплывающем меню пункт Revoke Publisher.
 - ◆ В папке SQL Remote щелкните правой кнопкой по пользователю, которому предоставлены полномочия PUBLISH, и выберите во всплывающем меню пункт Revoke Publisher.

Предоставление и отмена полномочий PUBLISH [Adaptive Server Anywhere]

Для Adaptive Server Anywhere полномочия PUBLISH предоставляются с помощью оператора GRANT PUBLISH:

```
GRANT PUBLISH TO идентификатор-пользователя ;
```

идентификатор-пользователя – это пользователь с полномочиями CONNECT в текущей базе данных. Например, следующий оператор предоставляет полномочия PUBLISH пользователю **S_Beaulieu**:

```
GRANT PUBLISH TO S_Beaulieu
```

Оператор REVOKE PUBLISH отменяет предоставление полномочий PUBLISH текущему издателю:

```
REVOKE PUBLISH FROM идентификатор-пользователя
```

Предоставление и отмена полномочий PUBLISH [Adaptive Server Enterprise]

Для Adaptive Server Enterprise полномочия PUBLISH предоставляются с помощью процедуры **sp_publisher**:

```
sp_publisher идентификатор-пользователя
```

идентификатор-пользователя – это пользователь с полномочиями CONNECT в текущей базе данных. Например, следующий оператор предоставляет полномочия PUBLISH пользователю **S_Beaulieu**:

```
exec sp_publisher 'S_Beaulieu'  
go
```

При выполнении процедуры **sp_publisher** без аргумента для базы данных устанавливается статус отсутствия издателя:

```
exec sp_publisher  
go
```

Примечания относительно полномочий PUBLISH

- ◆ Для просмотра идентификатора издателя для базы данных Adaptive Server Anywhere за пределами Sybase Central используйте специальную константу CURRENT PUBLISHER. Следующий оператор извлекает идентификатор **издателя**:

```
SELECT CURRENT PUBLISHER
```


- ◆ Для просмотра идентификатора издателя для базы данных Adaptive Server Enterprise используйте следующий оператор:

```
SELECT name
FROM sysusers
WHERE uid = ( SELECT user_id
              FROM sr_publisher )
go
```

- ◆ Если полномочия PUBLISH предоставлены пользователю с полномочиями GROUP, то они не наследуются членами группы.
- ◆ Полномочия PUBLISH используются только в целях идентификации издателя в исходящих сообщениях.
- ◆ Для отправляемых из текущей базы данных сообщений, которые будут получены и обработаны получателем, идентификатору издателя необходимо назначить полномочия REMOTE или CONSOLIDATE в базе данных получателя.
- ◆ Идентификатор издателя для базы данных не могут быть одновременно назначены полномочия REMOTE или CONSOLIDATE для этой базы данных, поскольку это приведет к идентификации данного издателя одновременно как отправителя исходящих сообщений и получателя этих сообщений.
- ◆ Изменение идентификатора издателя в удаленной базе данных может стать причиной серьезных проблем для всех подписок, в которых задействована эта база данных, включая потерю информации. Изменять идентификатор издателя удаленной базы данных не рекомендуется, так как это приведет к необходимости полной ресинхронизации удаленной базы данных.
- ◆ Изменение идентификатора издателя в консолидированной базе данных во время работы системы SQL Remote станет причиной серьезных проблем, включая потерю информации. Изменять идентификатор издателя консолидированной базы данных не рекомендуется, так как в противном случае потребуется завершить работу системы SQL Remote выполнить ресинхронизацию всех удаленных баз данных.

Предоставление и отмена полномочий REMOTE и CONSOLIDATE

Полномочия REMOTE и CONSOLIDATE очень сходны. Каждая база данных, получающая сообщения из текущей базы данных, должна иметь идентификатор пользователя, связанный с текущей базой данных, которому предоставлены полномочия REMOTE или CONSOLIDATE. Этот идентификатор пользователя представляет базу данных-получателя сообщений в текущей базе данных.

База данных, находящихся по иерархии SQL Remote непосредственно ниже текущей базы данных, предоставляются полномочия REMOTE, и по крайней мере одной базе данных выше текущей базы данных в иерархии предоставляются полномочия CONSOLIDATE.

Назначение полномочий REMOTE и CONSOLIDATE

В Adaptive Server Anywhere для идентификации системы передачи сообщений и указания адреса получателя сообщений репликации используются операторы GRANT REMOTE и GRANT CONSOLIDATE.

В Adaptive Server Enterprise для назначения полномочий REMOTE используется процедура `sp_grant_remote`, а для назначения полномочий CONSOLIDATE - процедура `sp_grant_consolidate`.

Полномочия CONSOLIDATE для консолидированной базы данных необходимо предоставлять даже удаленным базам данных, доступным только для чтения, поскольку осуществляется передача подтверждений получения сообщений от удаленных баз данных в консолидированную. Оператор GRANT CONSOLIDATE

в удаленных базах данных Adaptive Server Anywhere выполняется автоматически утилитой извлечения базы данных.

Предоставление полномочий REMOTE

Каждая удаленная база данных должна быть представлена в консолидированной базе данных одним идентификатором пользователя. Этому пользователю необходимо предоставить полномочия REMOTE для идентификации идентификатора пользователя и адреса как подписчика на публикации.

Полномочия REMOTE служат для выполнения несколько задач:

- ◆ Идентификация пользователя как удаленного пользователя;
- ◆ Определение используемого типа сообщений при обмене сообщениями с данным пользователем;
- ◆ Предоставление адреса получателя при отправке сообщений;
- ◆ Указание периодичности отправки сообщений удаленному пользователю.

Процедура предоставления полномочий REMOTE также называется добавлением удаленного пользователя в базу данных.

Пример для Sybase Central

Удаленного пользователя можно добавить в базу данных из Sybase Central. Удаленные пользователи и группы появляются в Sybase Central в двух местоположениях: в папке Users & Groups и в папке Remote Users (эти папки находятся в папке SQL Remote). В данном разделе рассматриваются только базы данных Adaptive Server Anywhere.

По умолчанию при создании удаленных пользователей и назначаются полномочия администратора удаленной БД. Поскольку этот тип полномочий требуется для получения доступа к удаленной базе данных из Message Agent, отменять эти полномочия нельзя.

Создание нового удаленного пользователя невозможно до тех пор, пока в базе данных определен хотя бы один тип сообщений.

При предоставлении полномочий REMOTE группе эти полномочия *не* предоставляются автоматически пользователям в группе (в отличие, к примеру, от полномочий для таблиц). Для этого необходимо явно предоставить полномочия REMOTE каждому пользователю в группе. В противном случае удаленные группы ведут себя точно так же, как и удаленные пользователи (и рассматриваются в системе как отдельные пользователи).

❖ Процедура добавления нового пользователя в базу данных в качестве удаленного пользователя (Sybase Central)

- 1 Откройте папку Remote Users (эта папка находится в папке SQL Remote).
- 2 Дважды щелкните по пункту Add Remote User и следуйте указаниям мастера.

❖ Процедура присвоения существующему пользователю статуса удаленного (Sybase Central)

- 1 Откройте папку Users & Groups.
- 2 Щелкните правой кнопкой по пользователю и выберите во всплывающем меню пункт Change to Remote User.
- 3 В открывающемся диалоге выберите из списка тип сообщений, введите адрес, выберите периодичность отправки сообщений и нажмите ОК для назначения пользователю статуса удаленного.

Пример для Adaptive Server Anywhere

Следующий оператор предоставляет полномочия REMOTE пользователю S_Beaulieu со следующими параметрами:

- ◆ Система отправки электронной почты SMTP;
- ◆ Отправка сообщений на адрес электронной почты **s_beaulieu@acme.com**;
- ◆ Периодичность отправки сообщений - ежедневно в 22:00.

```
GRANT REMOTE TO S_Beaulieu
TYPE smtp
ADDRESS 's_beaulieu@acme.com'
SEND AT '22:00'
```

Пример для Adaptive Server Enterprise

Следующий оператор предоставляет полномочия REMOTE пользователю **S_Beaulieu** со следующими параметрами:

- ◆ Для передачи сообщений используется система передачи сообщений путем совместного доступа к файлам;

- ◆ Сообщения размещаются в папке **beaulieu** в корневом адресном каталоге;

Корневой адресный каталог (для Adaptive Server Anywhere и для Adaptive Server Enterprise) указывается с помощью переменной среды SQLREMOTE (если она установлена). С другой стороны он указывается опцией Directory в параметрах управления сообщениями FILE (содержится в реестре или файле с расширением .INI).

- ◆ Периодичность отправки сообщений - каждые 12 часов:

```
exec sp_grant_remote 'S_Beaulieu',
'file',
'beaulieu',
'SEND EVERY',
'12:00'
go
```

Выбор периодичности отправки сообщений

Установить периодичность отправки сообщений можно с помощью следующих трех способов:

- ◆ **SEND EVERY.** Периодичность задается в часах, минутах и секундах в формате 'ЧЧ:ММ:СС'.

При отправке сообщений пользователю с установленными параметрами SEND EVERY сообщения также отправляются всем остальным пользователям с такими же настройками периодичности. Например, всем удаленным пользователям, получающим обновления с периодичностью "каждые двенадцать часов", обновления отправляются одновременно в указанное время 'ЧЧ:ММ:СС'. Это уменьшает количество обращений к журналу транзакций Adaptive Server Anywhere или очереди с сохранением Adaptive Server Enterprise. Возможность устанавливать различное время отправки сообщений рекомендуется использовать как можно реже.

- ◆ **SEND AT.** Время суток в часах и минутах.

Обновления запускаются ежедневно в заданное время. Для повышения эффективности рекомендуется использовать как можно меньше разных значений времени, а не назначать множество периодов отправки сообщений. Кроме того, для обеспечения возможности работы других пользователей время отправки сообщений должно приходиться на период сниженной нагрузки на базу данных.

- ◆ **Установка по умолчанию (отсутствие раздела SEND).** Если у какого-либо пользователя нет раздела SEND AT или SEND EVERY, то Message Agent отправляет сообщения всякий раз при запуске, после чего завершает свою работу (Message Agent использует режим пакетной обработки).

Установка периодичности отправки сообщений в Sybase Central

В Sybase Central периодичность отправки сообщений можно задать следующими способами:

- ◆ Во время присвоения существующему пользователю или группе статуса удаленного пользователя (группы). Для получения дополнительной информации см. раздел "Предоставление полномочий REMOTE" на стр. 178.
- ◆ На закладке Subscriptions окна свойств удаленного пользователя или группы. Окно свойств можно открыть путем щелчка правой кнопкой по удаленному пользователю или группе и выбором пункта Properties во всплывающем меню.

Предоставление полномочий CONSOLIDATE

В удаленной базе данных идентификатор издателя и идентификатор подписчика инвертированы (по сравнению с консолидированной базой данных). Подписчик (удаленный пользователь) в консолидированной базе данных становится издателем в удаленной базе данных. Издатель консолидированной базы данных становится подписчиком на публикацию из удаленной базы данных, и ему предоставляются полномочия CONSOLIDATE.

В каждой удаленной базе данных для консолидированной базы данных необходимо предоставить полномочия CONSOLIDATE. При создании удаленной базы данных с помощью утилиты извлечения базы данных оператор GRANT CONSOLIDATE автоматически выполняется в удаленной базе данных.

Пример для Adaptive Server Anywhere

Следующий оператор Adaptive Server Anywhere предоставляет полномочия CONSOLIDATE пользователю **hq_user** через систему электронной почты VIM:

```
GRANT CONSOLIDATE TO hq_user
TYPE vim
ADDRESS 'hq_address'
```

В этом операторе нет раздела SEND, поэтому используется значение по умолчанию, и сообщения будут отправляться в консолидированную базу данных каждый раз при запуске Message Agent.

Пример для Adaptive Server Enterprise

Следующий оператор Adaptive Server Enterprise предоставляет полномочия CONSOLIDATE пользователю **hq_user** через систему передачи сообщений:

```
exec sp_grant_consolidate 'hq_user', 'file', address
go
```

Отмена полномочий REMOTE и CONSOLIDATE

Пользователя можно удалить из системы SQL Remote путем отмены предоставления ему полномочий REMOTE. При отмене предоставления полномочий REMOTE пользователю или группе последние принимают статус "обычных". Этот пользователь или группа также автоматически исключаются из подписок на все публикации.

Отмена полномочий из Sybase Central

Отменить полномочия на базы данных Adaptive Server Anywhere можно из Sybase Central.

❖ Процедура отмены полномочий REMOTE (Sybase Central)

- 1 Откройте папку Users & Groups или Remote Users folder (эта папка находится в папке SQL Remote).
- 2 Щелкните правой кнопкой по удаленному пользователю или группе и выберите во всплывающем меню пункт Revoke Remote.

Отмена полномочий в Adaptive Server Anywhere

Предоставление пользователю полномочий REMOTE и CONSOLIDATE можно отменить с помощью оператора REVOKE. Следующий оператор отменяет полномочия REMOTE для пользователя **S_Beaulieu**.

```
REVOKE REMOTE FROM S_Beaulieu
```

Отмена полномочий
в Adaptive Server
Enterprise

Для отмены полномочий REMOTE или CONSOLIDATE необходимо обладать полномочиями администратора БД.

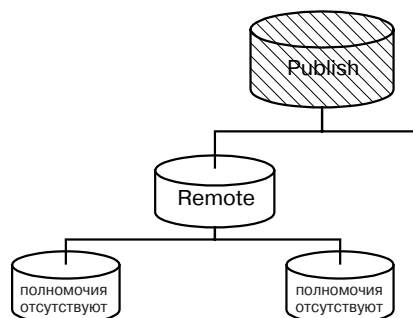
Предоставление пользователю полномочий REMOTE можно отменить при помощи процедуры **sp_revoke_remote**. Эта процедура использует один аргумент – идентификатор пользователя. Следующий оператор отменяет полномочия REMOTE для пользователя **S_Beaulieu**.

```
exec sp_revoke_remote 'S_Beaulieu'
go
```

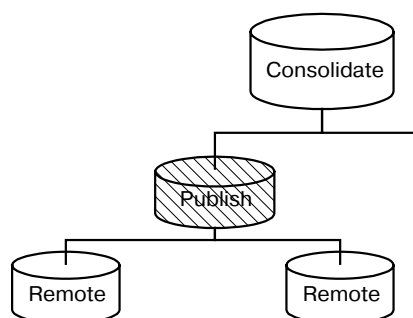
Назначение полномочий в многоуровневых системах

Назначение полномочий в многоуровневых системах требует особого рассмотрения. Полномочия в системе SQL Remote с тремя уровнями представлены в диаграммах ниже. На каждой диаграмме одна база данных затенена; диаграмма показывает полномочия, которые необходимо предоставить в этой базе данных идентификатору пользователя, представляющего одну из других баз данных. Фраза "Полномочия отсутствуют" означает, что базе данных в затененной базе данных не предоставлено никаких полномочий.

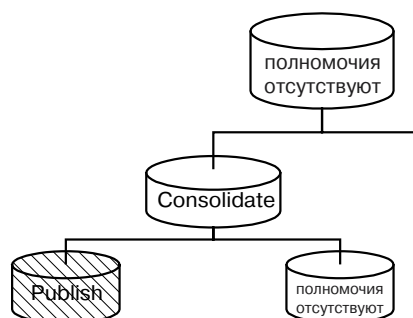
На следующем рисунке показаны полномочия SQL Remote, представленные в консолидированном узле трехуровневой системы.



На следующем рисунке показаны полномочия SQL Remote, представленные во внутреннем узле трехуровневой системы.



На следующем рисунке показаны полномочия SQL Remote, представленные во внутреннем узле трехуровневой системы.



Предоставление соответствующих полномочий PUBLISH и CONSOLIDATE в удаленных базах данных осуществляется автоматически утилитой извлечения базы данных.

Управление типами сообщений

SQL Remote поддерживает несколько различных систем передачи сообщений, а именно:

- ◆ **file.** Хранение файлов сообщений в каталогах файловой системы с совместным доступом, откуда они извлекаются другими базами данных.
- ◆ **ftp.** Хранение файлов сообщений в каталогах, доступных по каналу с протоколом передачи файлов (ftp).
- ◆ **mapi.** Система передачи сообщений API от Microsoft (MAPI), используемая в Microsoft Mail и других системах электронной почты.
- ◆ **smtp.** Простой протокол электронной почты Интернет (SMTP/POP), используемый в системах электронной почты в сети Интернет.
- ◆ **vim.** Система передачи сообщений между ПО независимых поставщиков от Lotus (VIM), используемая в Lotus Notes и cc:Mail.

База данных может осуществлять передачу сообщений с использованием одной или нескольких доступных систем передачи сообщений.

Доступность
операционной
системы

Не все системы передачи сообщений поддерживаются во всех операционных системах, с которыми совместим SQL Remote. В операционных системах Windows системы передачи сообщений реализуются как DLL.

☞ Список систем передачи сообщений, поддерживаемых той или иной операционной системой, см. в разделе "Поддерживаемые платформы и системы передачи сообщений" на стр. 449.

Дополнительная
информация

- ◆ Для получения дополнительной информации о системе передачи сообщений **file** см. раздел "Система передачи сообщений FILE" на стр. 187.
- ◆ Для получения дополнительной информации о системе передачи сообщений **ftp** см. раздел "Система передачи сообщений FTP" на стр. 188.
- ◆ Для получения дополнительной информации о системе передачи сообщений **smtp** см. раздел "Система передачи сообщений SMTP" на стр. 189.
- ◆ Для получения дополнительной информации о системе передачи сообщений **mapi** см. раздел "Система передачи сообщений MAPI" на стр. 191.
- ◆ Для получения дополнительной информации о системе передачи сообщений **vim** см. раздел "Система передачи сообщений VIM" на стр. 192.

Работа с типами сообщений

Каждое определение типа сообщений включает название типа (**file**, **ftp**, **smtp**, **mapi** или **vim**), а также адрес издателя для этого типа сообщений. Адрес издателя в консолидированной базе данных используется утилитой извлечения базы данных в качестве адреса возврата при создании удаленных баз данных. Адрес также используется в Message Agent для определения того, где располагаются входящие сообщения для системы **file**.

Адрес, предоставляемый вместе с определением типа сообщений, тесно связан с идентификатором издателя базы данных. Информация о допустимых адресах представлена в следующих разделах.

Перед использованием системы передачи сообщений необходимо задать адрес издателя.

Использование Sybase Central при работе с типами сообщений

В Sybase Central можно создавать и изменять типы сообщений. Типы сообщений представлены в папке Message Types (эта папка находится в папке SQL Remote). Информация в данном разделе относится только к базам данных Adaptive Server Anywhere.

Для создания и изменения типов сообщений необходимо обладать полномочиями администратора БД.

❖ Процедура добавления типа сообщений (Sybase Central)

- 1 Выполните подключение к базе данных.
- 2 Откройте папку SQL Remote этой базы данных.
- 3 В папке SQL Remote откройте папку Message Types.
- 4 Дважды щелкните по пункту Add Message Type.
- 5 В мастере создания типа сообщений (Message Type Creation) введите название типа сообщений. Это название должно соответствовать DLL типов сообщений, уже установленной в каталоге Adaptive Server Anywhere. Нажмите кнопку Next.
- 6 Введите адрес издателя и нажмите кнопку Finish для сохранения определения в базе данных.

При необходимости адрес издателя можно изменить путем изменения типа сообщений. Изменить название существующего типа сообщений нельзя; последний необходимо сначала удалить и создать новый тип сообщений с новым названием.

❖ Процедура изменения типа сообщений (Sybase Central)

- 1 Откройте папку SQL Remote для базы данных.
- 2 В папке SQL Remote откройте папку Message Types.
- 3 В правой области окна щелкните правой кнопкой по типу сообщений, который нужно изменить, и выберите во всплывающем меню пункт Properties.
- 4 В окне свойств сконфигурируйте требуемые параметры.

При необходимости тип сообщений можно удалить из системы.

❖ Процедура удаления типа сообщений (Sybase Central)

- 1 Откройте папку SQL Remote для базы данных.
- 2 В папке SQL Remote откройте папку Message Types.
- 3 В правой области окна щелкните правой кнопкой по типу сообщений, который нужно изменить, и выберите во всплывающем меню пункт Delete.

Создание типов сообщений для Windows CE

Из папки Sybase Central Utilities (при наличии установленных служб Windows CE) можно настроить SQL Remote для обеспечения синхронизации ActiveSync. При этом папка, назначенная как папка системы передачи сообщений FILE, становится папкой ActiveSync. При соединении компьютера Windows CE с пользовательским настольным компьютером ActiveSync обеспечивает синхронизацию файлов в папке ActiveSync настольного компьютера с файлами в папке Active Sync Windows CE.

Использование команд для работы с типами сообщений

❖ Процедура создания типа сообщений (SQL)

- 1 Выберите адрес издателя для данного типа сообщений.
- 2 Выполните оператор CREATE REMOTE MESSAGE TYPE.

Для Adaptive Server Anywhere оператор CREATE REMOTE MESSAGE TYPE имеет следующий синтаксис:

CREATE REMOTE MESSAGE TYPE *название-типа*
ADDRESS *строка-адреса*

Для Adaptive Server Enterprise используйте процедуру **sp_remote_type**. Эта процедура имеет следующие аргументы:

sp_remote_type *название-типа, строка-адреса*

В этих операторах *название-типа* - одна из систем передачи сообщений, поддерживаемых SQL Remote, а *строка-адреса* - адрес издателя в этой системе передачи сообщений.

При необходимости адрес издателя можно изменить путем изменения типа сообщений.

❖ Процедура изменения типа сообщений (SQL)

- 1 Выберите адрес издателя для данного типа сообщений.
- 2 Выполните оператор ALTER REMOTE MESSAGE TYPE.

Для Adaptive Server Anywhere оператор ALTER REMOTE MESSAGE TYPE имеет следующий синтаксис:

ALTER REMOTE MESSAGE TYPE *название-типа*
ADDRESS *строка-адреса*

Для Adaptive Server Enterprise пользуйтесь процедурой **sp_remote_type** так же, как и при создании типа сообщений. Эта процедура имеет следующие аргументы:

sp_remote_type *название-типа, строка-адреса*

В этих операторах *название-типа* - одна из систем передачи сообщений, поддерживаемая SQL Remote, а *строка-адреса* - адрес издателя в этой системе передачи сообщений.

Не используемые в системе репликации типы сообщений можно удалить. При этом из определения удаляется адрес издателя.

❖ Процедура удаления типа сообщений (SQL)

- ◆ Выполните оператор DROP REMOTE MESSAGE TYPE. Для Adaptive Server Anywhere

оператор DROP REMOTE MESSAGE TYPE имеет следующий синтаксис:

DROP REMOTE MESSAGE TYPE *название-типа*

Для Adaptive Server Enterprise пользуйтесь процедурой **sp_drop_remote_type** так же, как и при создании типа сообщений. Эта процедура имеет следующие аргументы:

sp_drop_remote_type *название-типа*

В этих операторах *название-типа* - одна из систем передачи сообщений, поддерживаемых SQL Remote.

☞ Для получения дополнительной информации см. также следующие разделы:

- ◆ "Оператор CREATE REMOTE MESSAGE TYPE" на стр. 306;
- ◆ "Оператор ALTER REMOTE MESSAGE TYPE" на стр. 304;
- ◆ "Оператор DROP REMOTE MESSAGE TYPE" на стр. 311.

Установка параметров управления типами сообщений

Каждая система передачи сообщений имеет несколько параметров, управляющих различными аспектами ее поведения. В разных системах передачи сообщений эти параметры различны, однако их установка осуществляется по тем же принципам.

При первом использовании Message Agent для определенной системы передачи сообщений Message Agent выводит диалоговое окно с набором параметров управления поведением системы. Этими параметрами может быть идентификатор пользователя для системы передачи сообщений, имя хоста, на котором хранятся сообщения ftp, и т.д. Вводимые значения параметров сохраняются в Message Agent. Эти параметры также можно задать явно.

Параметры канала передачи сообщений, хранящиеся в базе данных

Параметры управления сообщениями хранятся в базе данных. Задание этих параметров производится следующим образом:

❖ Процедура установки параметра управления сообщениями (Adaptive Server Anywhere)

- ◆ Выполните следующий оператор:

```
SET REMOTE имя-системы OPTION
[username.]имя-параметра = значение-параметра
```

❖ Процедура установки параметра управления сообщениями (Adaptive Server Enterprise)

- ◆ Выполните следующий оператор:

```
exec sp_link_option имя-системы, [имя-пользователя],
имя-параметра, значение-параметра
```

Текущие параметры системы передачи сообщений можно просмотреть с помощью запроса представления *sys.sysremotoptions* (Adaptive Server Anywhere) или представления *sr_remotoptions* (Adaptive Server Enterprise).

Хранение параметров канала передачи сообщений на диске

В более ранних версиях этого программного обеспечения параметры системы передачи сообщений хранились вне базы данных. Этим способом по-прежнему можно пользоваться, однако при отсутствии особых причин для внешнего хранения этих значений рекомендуется хранить параметры в базе данных.

Параметры управления системами передачи сообщений хранятся в следующих местоположениях:

- ◆ **Windows.** В реестре в следующем разделе:

```
\\HKEY_CURRENT_USER
  \Software
    \Sybase
      \SQL Remote
```

Параметры для каждой системы передачи сообщений хранятся в записи под записью SQL Remote с именем той или иной системы передачи сообщений (4, *smtip* и т.д.).

- ◆ **NetWare.** Необходимо создать файл с именем *dbremote.ini* в каталоге *sys:\system* для хранения установок каталога системы FILE. Формат этого файла отличается от формата .INI в Windows: он должен состоять из одной строки и содержать только имя папки.

Например, если это папка `user:\dbr43`, то файл `dbremote.ini` будет содержать следующее:

```
user:\dbr43
```

- ◆ **UNIX.** Установки каталога системы FILE хранятся в переменной среды SQLREMOTE.

В переменной среды `sqlremote` хранится путь, который можно использовать в качестве альтернативы для одного из параметров управления в системе передачи сообщений путем совместного доступа к файлам.

Параметры, доступные для каждой системы передачи сообщений, рассматриваются в следующих разделах. Каждый раздел посвящен одной системе передачи сообщений.

При загрузке системы передачи сообщений в Message Agent эта система использует параметры текущего издателя, либо, если параметр не задан, - групп, к которым принадлежит издатель. В Windows при первом запуске Message Agent версии с поддержкой хранения параметров системы передачи сообщений в базе данных, Message Agent копирует параметры системы из реестра в базу данных.

Система передачи сообщений FILE

SQL Remote можно использовать даже при отсутствии системы обмена сообщениями при помощи системы передачи сообщений **file**.

Адреса в системе передачи сообщений FILE

Система передачи сообщений **file** – простая система передачи сообщений путем совместного доступа к файлам. Адрес **file** для удаленного пользователя – это подкаталог, в который записываются все сообщения. Для извлечения сообщений из папки для входящих сообщений приложение считывает сообщения из каталога, содержащего файлы пользователя. Возвращаемые сообщения отправляются на адрес консолидированной базы данных (записанный для данной папки).

При запуске службы NT убедитесь, что для учетной записи, под которой работает Message Agent, имеются полномочия на чтение и запись для всех необходимых каталогов. Часто при доступе к сетевым дискам в этой связи возникают проблемы.

Корневой каталог для адресов

Обычно адреса системы **file** – это подкаталоги каталога, доступного для всех пользователей SQL Remote как через модем, так и через локальную сеть. Каждый пользователь должен иметь запись в реестре, запись в файле инициализации или переменную среды SQLREMOTE, указывающую каталог совместного доступа.

Систему **file** можно использовать для размещения сообщений в каталогах на компьютерах с консолидированной и удаленной базами данных. После этого простой механизм передачи файлов можно использовать для периодического обмена файлами в целях выполнения репликации.

Параметры управления сообщениями FILE

В системе передачи сообщений FILE используются следующие параметры управления:

- ◆ **Directory.** Установка каталога, в котором сохраняются сообщения. Данная установка является альтернативой переменной среде SQLREMOTE.
- ◆ **Debug.** Значения YES или NO (значение по умолчанию – NO). При установленном значении YES отображаются все вызовы файловой системы, выполненные системой FILE.

В NetWare необходимо создать файл с именем `dbremote.ini` в каталоге `sys:\system` для сохранения установок каталога.

Система передачи сообщений FTP

Адреса для FTP

В системе передачи сообщений FTP сообщения хранятся в папках корневого каталога на хосте FTP. Хост FTP и корневой каталог задаются путем установки параметров управления системой передачи сообщений, хранящихся в реестре или файле инициализации. При этом адрес каждого пользователя является подкаталогом, в котором хранятся сообщения данного пользователя.

☞ Список операционных систем с поддержкой FTP см. в разделе "Поддерживаемые операционные системы" на стр. 391.

Параметры управления сообщениями FTP

В системе передачи сообщений FTP используются следующие параметры управления:

- ◆ **host.** Имя хоста компьютера, на котором хранятся сообщения. Это может быть имя хоста (например, **ftp.ianywhere.com**) или IP-адрес (например, 192.138.151.66).
- ◆ **user.** Имя пользователя для доступа к хосту FTP.
- ◆ **password.** Пароль для доступа к хосту FTP.
- ◆ **root_directory.** Корневой каталог на узле хоста FTP, в котором хранятся сообщения.
- ◆ **port.** Обычно не требуется. Номер IP-порта, используемого для подключения к FTP.
- ◆ **debug.** Значения YES или NO (значение по умолчанию – NO).
При установленном значении YES отображается результат отладки.
- ◆ **active_mode** Значения YES или NO (значение по умолчанию – NO) (пассивный режим).

Устранение неисправностей FTP

Большинство проблем с системой передачи сообщений FTP связано с сетевыми настройками. В данном разделе представлен перечень тестов, которые можно провести с целью устранения неисправностей.

Задайте параметр управления сообщениями DEBUG. Просмотр результата отладки должен определить наличие или отсутствие подключения к серверу FTP. При наличии подключения будут указаны невыполненные FTP-команды.

Эхо-тестирование (ping) сервера FTP Если канал FTP не подключается к серверу FTP, попробуйте проверить системную конфигурацию сети. Если в системе поддерживается команда эхо-тестирования (**ping**), попробуйте ввести следующую команду:

```
ping имя-сервера-ftp
```

В результате выполнения команды выводится IP-адрес сервера и время эхо-тестирования (время отправки запроса и получения ответа) сервера. Если при наличии проблем конфигурации сети провести эхо-тестирование сервера нельзя, обратитесь к администратору сети.

Проверка работы пассивного режима. Если канал FTP подключается к серверу FTP, но не может установить соединение для передачи данных, проверьте, может ли клиент FTP использовать пассивный режим для обмена данными с сервером.

Пассивный режим является предпочтительным режимом передачи и устанавливается по умолчанию для системы передачи сообщений FTP. В пассивном режиме все соединения для передачи данных инициируются клиентом, в данном случае – системой передачи сообщений. В активном режиме все соединения для передачи данных инициируются сервером. Если сервер FTP находится за неправильно сконфигурированным брандмауэром, возможно, что

использовать устанавливаемый по умолчанию режим пассивной передачи будет нельзя. В этой ситуации брандмауэр блокирует разъемы подключения к серверу FTP на портах, отличных от порта управления FTP.

С помощью пользовательского ПО доступа к FTP, позволяющего задавать **активный** или **пассивный** режим передачи, установите режим передачи на пассивный и попробуйте выгрузить или загрузить какой-нибудь файл. Если используемый клиент не может передать файл без использования активного режима, тогда нужно либо повторно сконфигурировать брандмауэр и сервер FTP для обеспечения возможности передачи в пассивном режиме, либо установить параметр управления *active_mode* в YES. Передача в активном режиме поддерживается не во всех сетевых конфигурациях. Например, если клиент находится за шлюзом, имитирующем IP, тогда входящие подключения могут не состояться (это зависит от программного обеспечения шлюза).

Проверка полномочий и структуры каталога. Если подключение к серверу FTP выполняется, но возникают проблемы с получением списков каталогов или при работе с файлами, проверьте правильность настроек полномочий и наличие необходимых каталогов.


Войдите на сервер FTP с помощью ПО доступа к FTP. Перенесите каталоги в местоположение, указанное в параметре *root_directory*. Если необходимые каталоги не отображаются, то это означает либо неправильную установку параметра управления *root_directory*, либо отсутствие данных каталогов.

Проверьте наличие необходимых полномочий путем выбора файла в каталоге сообщений и его загрузки в консолидированную базу данных. Если возникает ошибка, это означает неправильную настройку полномочий сервера FTP.

Система передачи сообщений SMTP

Простой протокол электронной почты (Simple Mail Transfer Protocol, SMTP) используется в программных продуктах электронной почты в сети Интернет.

С помощью SMTP SQL Remote осуществляет передачу сообщений через почтовые системы сети Интернет. Сообщения шифруются в текстовый формат и отправляются в виде сообщений электронной почты в нужную базу данных. Отправка сообщений осуществляется через сервер SMTP, а получение – через сервер POP: по такой схеме отправка и получение сообщений производится во многих программах электронной почты.

 Список операционных систем с поддержкой SMTP см. в разделе "Поддерживаемые операционные системы" на стр. 391.

Адреса SMTP и идентификаторы пользователя

Для работы с SQL Remote и системой передачи сообщений SMTP каждой базе данных, участвующей в обмене сообщениями, необходим адрес SMTP, идентификатор пользователя POP3 и пароль. Это различные идентификаторы: адрес SMTP – конечная цель каждого сообщения; идентификатор пользователя POP3 и пароль - имя и пароль, вводимые пользователем при подключении к своему почтовому ящику.

Рекомендуется создать отдельную учетную запись для электронной почты

Для сообщений SQL Remote рекомендуется иметь отдельную учетную запись электронной почты POP.

Устранение неисправностей

Если канал SMTP не работает, попробуйте подключиться к серверу SMTP/POP3 с компьютера, на котором запущен Message Agent, с использованием той же учетной записи и пароля. Воспользуйтесь программой электронной почты Интернет, поддерживающей SMTP/POP3, и обязательно закройте эту программу после восстановления рабочего состояния системы передачи сообщений SMTP.

Параметры управления системой передачи сообщений SMTP

Перед тем, как Message Agent начнет процесс установления соединения с системой передачи сообщений для их отправки или получения, на компьютере пользователя уже должен быть установлен набор параметров управления, либо пользователь должен ввести в специальное окно требуемую информацию. Эту информацию необходимо вводить только при первом подключении. Она сохраняется и при последующих подключениях используется по умолчанию.

В системе передачи сообщений SMTP используются следующие параметры управления:

- ◆ **local_host.** Это имя локального компьютера. Данный параметр используется на компьютерах, где SQL Remote не может определить имя локального хоста. Имя локального хоста необходимо для инициализации сеанса связи с любым сервером SMTP. В большинстве сетевых сред имя локального хоста может определяться автоматически, и необходимости в этой записи нет.
- ◆ **TOP_supported.** При перечислении входящих сообщений SQL Remote использует команду POP3 с именем TOP. Команда TOP не обязательно поддерживается всеми серверами POP. При установке этого параметра в NO будет использоваться команда RETR, которая менее эффективна, но поддерживается всеми серверами POP. Значение по умолчанию - YES.
- ◆ **smtp_authenticate.** Определяет, выполняется ли аутентификация пользователя в системе SMTP. Значение по умолчанию - YES. При установке в NO аутентификация SMTP осуществляться не будет.
- ◆ **smtp_userid.** Идентификатор пользователя для аутентификации SMTP. По умолчанию этот параметр принимает то же значение, что и параметр **pop3_userid**. Если идентификатор пользователя отличается от указанного на сервере POP, то необходимо задать только **smtp_userid**.
- ◆ **smtp_password.** Пароль для аутентификации SMTP. По умолчанию этот параметр принимает то же значение, что и параметр **pop3_password**.
Если идентификатор пользователя отличается от указанного на сервере POP, то необходимо задать только **smtp_password**.
- ◆ **smtp_host.** Имя компьютера, на котором запущен сервер SMTP. Оно соответствует установке хоста SMTP в диалоговом окне входа в систему SMTP/POP3.
- ◆ **pop3_host.** Имя компьютера, на котором запущен хост POP. Обычно это имя совпадает с именем хоста SMTP. Это имя соответствует установке хоста POP3 в диалоговом окне входа в систему SMTP/POP3.
- ◆ **pop3_userid.** Используется для получения почты. Идентификатор пользователя POP соответствует установке идентификатора пользователя в диалоговом окне входа в систему SMTP/POP3. Идентификатор пользователя можно получить у администратора хоста POP.
- ◆ **pop3_password.** Используется для получения почты. Соответствует установке пароля в диалоговом окне входа в систему SMTP/POP3. Если все пять вышеуказанных параметров установлены, то диалог входа в систему не отображается.
- ◆ **Debug.** При установке в YES отображается информация обо всех посланных командах SMTP и POP3 и полученных ответах. Это может быть полезно при устранении неисправностей SMTP/POP. Значение по умолчанию - NO.

Совместное использование адресов SMTP/POP

База данных должна иметь собственную учетную запись электронной почты для сообщений SQL Remote для их получения отдельно от персональных сообщений, предназначенных для чтения. Это необходимо потому, что многие адресаты сообщений будут получать их следующим способом:


- 1 Подключение к хосту POP и загрузка всех сообщений;
- 2 Удаление всех сообщений с хоста POP;
- 3 Отключение от хоста POP;
- 4 Прочтение почты из локального файла или из памяти.

Это может вызывать проблемы, поскольку почтовая программа загружает и удаляет все сообщения электронной почты SQL Remote наряду с персональными сообщениями. При обеспечении того, что почтовая программа не удалит непрочитанные сообщения с хоста POP, можно использовать адрес электронной почты совместно с базой данных до тех пор, пока сообщения базы данных не удаляются или в них не вносятся изменения.

Эти сообщения легко распознать, поскольку они заполнены строками на первый взгляд лишенных смысла символов.

Система передачи сообщений MAPI

Интерфейс прикладного программирования для электронной почты (Message Application Programming Interface, MAPI) используется в нескольких популярных почтовых системах, например, Microsoft Mail и в более поздних версиях Lotus cc:Mail.

 Список операционных систем с поддержкой MAPI см. в разделе "Поддерживаемые операционные системы" на стр. 391.

Адреса MAPI и идентификаторы пользователя

Для работы с SQL Remote и системой передачи сообщений MAPI каждой базе данных, участвующей в обмене сообщениями, необходим идентификатор пользователя MAPI и адрес. Это различные идентификаторы: адрес MAPI – конечная цель каждого сообщения; идентификатор пользователя MAPI и пароль – имя и пароль, вводимые пользователем при подключении к своему почтовому ящику.

Сообщения MAPI и почтовый ящик для входящих сообщений

Хотя сообщения SQL Remote могут приходиться в тот же почтовый ящик, что и сообщения электронной почты, предназначенные для чтения, они, как правило, не отображаются в ящике для входящих сообщений.

SQL Remote отправляет сообщения с указанием приложения для их обработки, которые MAPI идентифицирует и скрывает при открытии почтового ящика. Таким образом, пользователи могут использовать тот же адрес электронной почты и то же подключение для получения как персональной почты, так и обновлений базы данных, и при этом сообщения SQL Remote не будут смешиваться с почтой, предназначенной для чтения.

Если сообщение направлено через Интернет, тогда особая информация о типе сообщения будет потеряна. В этом случае эти сообщения отображаются в ящике для входящих сообщений.

Параметры управления системы передачи сообщений MAPI

В системе передачи сообщений MAPI используются следующие параметры управления:

- ◆ **Debug.** При установке в YES выводятся все вызовы MAPI и коды возврата. Это может быть полезно при устранении неисправностей MAPI. Значение по умолчанию - NO.
- ◆ **Force_Download.** (по умолчанию YES) Управление включением флага MAPI_FORCE_DOWNLOAD при вызове MapiLogon. Данный параметр может быть полезным при использовании удаленного почтового программного обеспечения, которое при включенном флаге начинает процесс установки соединения.

- ◆ **IPM_Receive.** Значения YES или NO (по умолчанию – NO). При установке в YES система MAPI получает сообщения IPM, которые отображаются в почтовом ящике. При установке в NO система MAPI получает сообщения IPC, которые не отображаются в почтовом ящике. Данный параметр может быть полезен, если провайдер MAPI не поддерживает сообщения IPC. Также параметр может быть полезен при приеме сообщений через Интернет. В этом случае отправитель, возможно, не использовал MAPI, либо атрибуты IPC были утеряны.
- ◆ **IPM_Send.** Значения YES или NO (по умолчанию - NO). При установке в YES система MAPI отправляет сообщения IPM, которые отображаются в почтовом ящике. При установке в NO система MAPI отправляет сообщения IPC, которые не отображаются в почтовом ящике. Данный параметр может быть полезен, если провайдер MAPI не поддерживает сообщения IPC.
- ◆ **Profile.** Использование заданного профиля Microsoft Exchange. Применяется, если Message Agent функционирует в качестве службы.

Система передачи сообщений VIM

Система передачи сообщений между ПО независимых поставщиков (Vendor Independent Messaging system, VIM) используется в Lotus Notes и в некоторых версиях Lotus cc:Mail.

Для работы с SQL Remote и системой передачи сообщений VIM каждой базе данных, участвующей в обмене сообщениями, необходимо назначить идентификатор пользователя VIM и адрес. Это различные идентификаторы: адрес VIM – конечная цель каждого сообщения; идентификатор пользователя VIM – имя, вводимое пользователем при подключении к своему почтовому ящику.

☞ Список операционных систем с поддержкой VIM см. в разделе "Поддерживаемые операционные системы" на стр. 391.

Параметры
управления системой
передачи сообщений
VIM

В системе передачи сообщений VIM используются следующие параметры управления:

- ◆ **Path.** Соответствует установке Path в диалоге входа в систему cc:Mail. С Lotus Notes не применяется и игнорируется.
- ◆ **Userid.** Соответствует установке User ID в диалоге входа в систему cc:Mail.
- ◆ **Password.** Соответствует установке Password в диалоге входа в систему cc:Mail. Если заданы параметры Path, Userid и Password, тогда диалог входа в систему не отображается.
- ◆ **Debug.** При установке в YES отображаются все вызовы VIM и коды возврата. Данный параметр полезен при устранении неисправностей VIM. Значение по умолчанию - NO.
- ◆ **Receive_All.** При установке в YES Message Agent проверяет все сообщения на предмет их принадлежности к SQL Remote. При установке в NO (значение по умолчанию) Message Agent просматривает только сообщения типа **SQLRemoteData** (указание приложения для обработки). Это значительно повышает производительность Notes.

Установка параметра **ReceiveAll** в YES может использоваться в случаях, где тип сообщений потерян, удален или не был задан вообще. Это относится к сообщениям cc:Mail или сообщениям, передаваемым через Интернет.

- ◆ **Send_VIM_Mail.** При установке в YES Message Agent отправляет сообщения, совместимые с версиями Adaptive Server Anywhere до 5.5.01 и одновременно совместимые с cc:Mail. При установке в YES необходимо проверить, что параметр **Receive_All** также установлен в YES.

Работа с Message Agent

SQL Remote Message Agent - ключевой компонент в системе репликации SQL Remote. Message Agent управляет как отправкой, так и получением сообщений. Он выполняет следующие функции:

- ◆ Message Agent обрабатывает входящие сообщения и располагает их в надлежащем порядке в базе данных.
- ◆ Message Agent сканирует журнал транзакций или очередь с сохранением в каждой базе данных издателя и переводит записи журнала в сообщения для подписчиков.
- ◆ Message Agent компонует пакеты записей журнала в сообщения размером не больше максимального заданного (50 000 байт по умолчанию) и отправляет их подписчикам.
- ◆ Также он предоставляет информацию по отслеживанию сообщений в системных таблицах и обеспечивает безотказный механизм передачи.

Имена исполняемых файлов

В операционных системах Windows Message Agent для Adaptive Server Enterprise называется *ssremote.exe*, а Message Agent для Adaptive Server Anywhere - *dbremote.exe*. В операционных системах UNIX это имена *ssremote* и *dbremote* соответственно.

☞ Message Agent для Adaptive Server Enterprise использует очередь с сохранением для хранения транзакций до тех пор, пока необходимость в них не отпадет. Для получения дополнительной информации об очереди с сохранением см. раздел "Принципы работы Message Agent для Adaptive Server Enterprise" на стр. 230.

Режимы пакетной передачи и непрерывной работы в Message Agent

Message Agent можно запустить в одном из двух режимов:

- ◆ **Режим пакетной передачи.** В режиме пакетной передачи Message Agent запускается, принимает и отправляет все сообщения, имеющиеся для приема и отправки, после чего завершает свою работу.

Режим пакетной передачи используется на удаленных узлах с непостоянным подключением, где обмен сообщениями может происходить только с консолидированной базой во время сеанса связи, например, когда удаленный пользователь набирает номер для выхода в основную сеть.

- ◆ **Режим непрерывной работы.** В режиме непрерывной работы Message Agent периодически отправляет сообщения в соответствии со временем, указываемым в свойствах каждого удаленного пользователя. При отсутствии отправляемых сообщений осуществляется прием входящих сообщений по мере их поступления.

Режим непрерывной работы используется на консолидированных узлах, откуда сообщения могут отправляться и куда они могут поступать в любой момент времени, что позволяет распределить рабочую нагрузку и обеспечить непрерывную репликацию.

Доступные параметры зависят от параметров периодичности отправки сообщений, заданных для удаленных пользователей. Параметры периодичности отправки сообщений описаны в разделе "Выбор периодичности отправки сообщений" на стр. 179.

❖ **Процедура запуска Message Agent в режиме непрерывной работы**

- 1 Убедитесь, что для каждого пользователя задана периодичность отправки сообщений. Периодичность отправки задается параметром SEND AT или SEND EVERY в операторе GRANT REMOTE (Adaptive Server Anywhere) или процедуре **sp_grant_remote** (Adaptive Server Enterprise).
- 2 Запустите Message Agent без параметра -b.

❖ **Процедура запуска Message Agent в режиме пакетной передачи**

- ◆ Необходимо следующее:
 - ◆ наличие как минимум одного удаленного пользователя без установленных параметров SEND AT или SEND EVERY в свойствах, либо
 - ◆ запуск Message Agent с параметром -b.

Подключения, используемые Message Agent

Message Agent использует несколько подключений к серверу базы данных. Это следующие подключения:

- ◆ Глобальное подключение, которое активизируется все время работы Message Agent.
- ◆ Подключение для сканирования журнала. Это подключение активизируется только во время сканирования.
- ◆ Подключение для выполнения команд потока сканирования журнала. Это подключение активизируется только во время сканирования.
- ◆ Подключение для очереди с сохранением (только для Adaptive Server Enterprise). Это подключение активизируется только во время сканирования и отправки сообщений.
- ◆ Подключение для обработки запросов на синхронизацию подписки. Это подключение активизируется только во время отправки сообщений.
- ◆ Подключение для каждого рабочего потока. Эти подключения активизируются только во время приема сообщений.

Процедуры восстановления системы репликации

К методам восстановления данных в узлах консолидированной базы данных система репликации SQL Remote предъявляет новые требования. Стандартное резервное копирование и соответствующие процедуры обеспечивают восстановление данных после отказов системы или носителей. Даже после успешного восстановления в системе репликации может быть нарушена синхронизация восстановленной базы с удаленными базами данных. В этой связи может потребоваться полная повторная синхронизация удаленных баз данных (задача не из простых, если в системе репликации задействовано большое количество баз данных).

Одним словом, восстановление консолидированной базы данных после сбоя в консолидированном узле – всего лишь часть задачи восстановления всей системы репликации.

Защита системы репликации от отказов носителей реализуется с помощью следующих двух способов:

- ◆ **Резервное копирование и управление журналом.** Расширенные процедуры резервного копирования, а также процедуры управления журналом для сервера консолидированной базы данных представляют собой существенную часть операций в плане восстановления. Процедуры резервного копирования обеспечивают защиту от отказа носителей на устройстве хранения базы данных. Использование зеркальной копии журнала транзакций обеспечивает защиту от отказа носителей на устройстве журнала транзакций.

☞ Для получения дополнительной информации о резервном копировании и процедурах управления журналом см. разделы "Управление журналом транзакций и резервным копированием" на стр. 216 и "Управление журналом транзакций и резервным копированием Adaptive Server Enterprise" на стр. 281.

- ◆ **Конфигурирование Message Agent.** Параметры командной строки Message Agent обеспечивают возможность подстройки поведения Message Agent под требования задач резервного копирования и восстановления.

Вопросы конфигурирования Message Agent рассматриваются ниже.

Репликация только транзакций с резервными копиями

По умолчанию Message Agent обрабатывает все выполненные транзакции. При запуске Message Agent с параметром `-u` обрабатываются только те транзакции, резервные копии которых были созданы при помощи команд резервного копирования базы данных.

Для Adaptive Server Anywhere резервное копирование журнала транзакций выполняется при помощи Sybase Central или утилиты `dbbackup`, либо копированием и переименованием файла журнала транзакций в режиме off-line. Для Adaptive Server Enterprise резервное копирование журнала транзакций выполняется с помощью оператора дампа транзакций (**dump transaction**).

Путем репликации только транзакций с резервными копиями система репликации обеспечивает защиту от отказов носителей журнала транзакций. Создание зеркальной копии журнала транзакций также способствует достижению этой цели.

Параметр `-u` обеспечивает дополнительную защиту от полного отказа узла в случае, если резервное копирование выполняется на другом узле.

Обеспечение согласованности параметров Message Agent

Некоторые параметры Message Agent должны быть одинаковыми в пределах всей системы репликации, поэтому их необходимо задать до начала развертывания последней. В данном разделе перечислены эти параметры.

- ◆ **Максимальная длина сообщения.** По умолчанию максимальная длина сообщений SQL Remote - 50 Кб. Это значение можно изменить с помощью параметра `-l` Message Agent. Однако максимальная длина сообщений должна быть одинаковой для каждого Message Agent в системе репликации; ее можно ограничить пределами выделения памяти операционной системы.

Полученные сообщения, превышающие по длине заданный предел, удаляются как внеочередные.

☞ Для получения подробной информации об этом параметре см. раздел "Message Agent" на стр. 9.

Безопасность сообщений Message Agent и системы репликации

Сообщения, отправляемые SQL Remote Message Agent, шифруются с помощью простого ключа, что защищает их от случайного просмотра. Однако применяемая

схема шифрования не обеспечивает полной защиты от намеренных попыток их раскодирования.

Устранение неисправностей в удаленных узлах

Для администратора, имеющего доступ только к консолидированному узлу, существуют очевидные препятствия для устранения ошибок, возникающих в удаленных узлах. Для решения этой проблемы SQL Remote можно настроить так, чтобы некоторые данные их журнала исходящих сообщений удаленных узлов поступали в консолидированный узел и записывались в файл. Этот файл будет содержать информацию журнала из некоторых или всех узлов системы.

Для поддержки функции сбора информации журнала в SQL Remote необходимо сконфигурировать как удаленные, так и консолидированные узлы.

❖ Процедура конфигурирования удаленной базы данных для отправки информации журнала в консолидированную базу данных

1. Задайте параметр системы передачи сообщений для отправки информации журнала при возникновении ошибки. Выполните следующую команду в удаленной базе данных:

```
SET REMOTE ИМЯ-СИСТЕМЫ OPTION  
PUBLIC.OUTPUT_LOG_SEND_ON_ERROR = 'YES'
```

После установки этого параметра любое сообщение, начинающееся с индикатора ошибки 'E', вынуждает SQL Remote отправлять информацию журнала на консолидированный узел.

☞ Для получения дополнительной информации см. раздел "Оператор SET REMOTE OPTION [SQL Remote]" (SET REMOTE OPTION statement [SQL Remote]) на стр. 543 в документе "*Справочник по SQL для ASA*" (*ASA SQL Reference Manual*).

2. Задайте параметр системы передачи сообщений так, чтобы ограничить объем информации, отправляемой на консолидированный узел. Этот шаг является необязательным.

Выполните следующую команду в удаленной базе данных:

```
SET REMOTE ИМЯ-СИСТЕМЫ OPTION  
PUBLIC.OUTPUT_LOG_SEND_LIMIT = 'nnn'
```

Значение этого параметра - число байт в заключительной части журнала исходящий сообщений (т.е. в самых последних записях), отправляемого в консолидированный узел. Для указания "килобайт" можно использовать обозначение *nnnK*. Значение по умолчанию - '5K' (5 Кб).

При вводе значения, превышающего максимальный размер сообщения, SQL Remote заменяет данное значение параметра и отправляет информацию в объеме, не превосходящем размеры сообщения.

Информацию журнала можно отправлять даже при отсутствии ошибок путем установки параметра OUTPUT_LOG_SEND_NOW в YES. SQL Remote отправляет информацию журнала исходящих сообщений при следующем опросе и после отправки журнала устанавливает данный параметр в "NO".

❖ Процедура конфигурирования консолидированного узла для получения информации журнала

- ◆ Используйте параметр `-ro` или `-rt` Message Agent.

☞ Для получения дополнительной информации см. раздел "Message Agent" на стр. 9.

Повышение производительности Message Agent

Для кого предназначен данный раздел? Если производительность узла полностью устраивает пользователя, тогда данный раздел можно пропустить.

Существует несколько вариантов повышения производительности Message Agent. Эти варианты описываются в данном разделе.

Отправка и получение сообщений – это два разных процесса. Поэтому основные методы повышения производительности для этих двух процессов различны.

- ◆ **Производительность при репликации.** Основной проблемой для общей производительности узлов SQL Remote, как правило, является получение сообщений со многих удаленных баз данных и их обработка в базе данных в консолидированном узле. Этими процессами можно управлять путем настройки процесса приема Message Agent на консолидированном узле.
- ◆ **Задержка при репликации.** Период времени с момента поступления данных на один узел до момента их появления на других узлах является временем задержки при репликации. Этой задержкой также можно управлять.

Повышение производительности путем управления многопоточностью Message Agent

В данном разделе предполагается выполнение пользователем действий по повышению производительности Message Agent, работающего в режиме непрерывной работы на консолидированном узле.

Message Agent может использовать рабочие потоки для обработки входящих сообщений от удаленных пользователей. Это может повысить производительность благодаря параллельной (а не последовательной) обработке сообщений.

Установка количества рабочих потоков

Количество рабочих потоков задается в командной строке Message Agent при помощи параметра `-w`. Следующая командная строка запускает Message Agent для Adaptive Server Enterprise с двадцатью рабочими потоками обработки сообщений:

```
ssremote -c "eng=..." -w 20
```

По умолчанию рабочие потоки не используются вообще, поэтому все сообщения обрабатываются последовательно. Максимальное количество рабочих потоков – 50.

Повышение производительности при применении рабочих потоков

Для Message Agent для Adaptive Server Anywhere повышение производительности будет наибольшим в случае, если сервер расположен в системе с распределенным дисковым массивом.

Для Adaptive Server Enterprise Message Agent получит еще большее преимущество, если сервер используется в конфигурации с множественными процессорами.

Сообщения, обрабатываемые параллельно

При использовании рабочих потоков сообщения от разных удаленных пользователей обрабатываются параллельно. Сообщения от одного удаленного пользователя обрабатываются последовательно. Например, десять сообщений от одного удаленного пользователя будут обработаны одним рабочим потоком в правильном порядке.

В случае взаимной блокировки применяется повторная обработка возвращаемой транзакции в более позднее время.

Прочтение сообщений из системы передачи сообщений является процессом с одним потоком. Перед отправкой сообщений в рабочие потоки для обработки

сообщения считываются, и проверяется информация заголовка (для определения удаленного пользователя и правильного порядка обработки сообщений).

Процесс формирования и отправки сообщений является процессом с одним потоком.

Версия Open Client

Для использования множественных рабочих потоков с Message Agent Adaptive Server Enterprise необходим Open Client версии 11.1 или более поздней.

При работе с версиями до 11.1 Message Agent распечатывает сообщение, после чего не использует рабочие потоки. Версия Open Client отображается в первых нескольких строках выводимых данных Message Agent.

Повышение производительности путем кэширования сообщений

Message Agent кэширует входящие сообщения в конфигурируемой области памяти по мере их считывания.

Определение размера кэша сообщений

Размер кэша сообщений задается в командной строке Message Agent с помощью параметра `-m`.

Параметр `-m` задает максимальный объем памяти, который используется Message Agent для формирования сообщений. Допустимый размер можно задать как *n* (в байтах), *nK* или *nM*. Значение по умолчанию - 2048 Кб (2 Мб).

Пример

Следующая командная строка запускает Message Agent в Adaptive Server Anywhere при использовании 12 Мб памяти в качестве кэша сообщений:

```
dbremote -c "eng=..." -m 12M
```

Кэширование сообщений

Если транзакции слишком большие или сообщения поступают не в заданном порядке, тогда Message Agent сохраняет их в памяти до тех пор, пока сообщение не будет обработано. Такое кэширование сообщений предотвращает повторное считывание внеочередных сообщений из системы передачи сообщений, что может отрицательно повлиять на производительность в крупных системах репликации. Это особенно важно, когда сообщения считываются по WAN (например, услуги удаленного доступа или POP3 через модем). При этом пользователь избегает возникновения конфликтов между рабочими потоками, считывающими сообщения (однопоточное задание), поскольку содержание сообщения кэшировано.

При превышении выделенного объема памяти, заданного при помощи параметра `-m`, сообщения удаляются, начиная с сообщений, использованных наиболее давно.

В основном этот параметр предназначен для систем, в которых одна консолидированная база данных используется для тысяч удаленных баз данных.

Настройка опроса входящих сообщений

При работе с Message Agent в режиме непрерывной работы (как правило, на узле консолидированной базы данных) пользователь может управлять процессом опроса входящих сообщений в Message Agent, а также временем ожидания сообщений Message Agent, поступающих не в заданном порядке, перед выполнением запроса о повторной передаче сообщения. В некоторых ситуациях настройка этих аспектов поведения может существенно влиять на производительность.

Положения, которые необходимо учитывать

Положения, которые необходимо принимать во внимание во время настройки процесса приема сообщений, схожи с аспектами настройки процесса отправки сообщений.

- ◆ **Обычные сообщения.** Пользователь может настроить периодичность опроса Message Agent входящих сообщений из удаленных баз данных.
- ◆ **Запросы на повторную передачу.** Пользователь может контролировать количество ожидаемых опросов перед поступлением внеочередного сообщения, перед выполнением запроса о повторной передаче данного сообщения.
- ◆ **Обработка входящих сообщений.** Если период опроса для входящих сообщений слишком продолжителен по сравнению с частотой их поступления, может возникнуть ситуация постоянного нахождения сообщений в очереди на обработку. Если период опроса слишком мал, это приведет к неэффективному расходу ресурсов при выполнении опроса при отсутствии сообщений в очереди.

☞ Для получения дополнительной информации о процессе отправки сообщений см. раздел "Настройка процесса отправки сообщений" на стр. 201.

Период опроса

По умолчанию Message Agent, работающий в режиме непрерывной работы, выполняет опрос для проверки поступления новых сообщений спустя минуту после завершения предыдущего опроса. Период опроса можно сконфигурировать при помощи параметра `-rd`.

Период опроса по умолчанию, начиная с конца одного сеанса опроса до начала следующего, – 1 минута. При указании значения в секундах, как показано в командной строке ниже, опрос можно выполнять чаще:

```
dbremote -rd 30s
```

Точно так же опрос можно выполнять реже (см. следующую командную строку), например, каждые 5 минут:

```
dbremote -rd 5
```

Указание слишком короткого интервала может оказать неблагоприятное воздействие на общую производительность системы по следующим причинам:

- ◆ Каждый опрос почтового сервера (при работе с электронной почтой) создает нагрузку на систему передачи сообщений. Слишком частые опросы могут повлиять на работу системы передачи сообщений без получения каких-либо преимуществ.
- ◆ Если пользователь не изменит время ожидания Message Agent до того, как последний предположит, что сообщение, поступившее вне заданной последовательности, утеряно, после чего направит запрос о повторной отправке этого сообщения, тогда существует опасность переполнения системы запросами на повторную передачу.

В общем случае, использовать слишком короткий интервал опроса не рекомендуется, если на это нет каких-то особых причин, по которым необходимо установить очень короткое время реакции на поступившие сообщения.

Установка более продолжительных периодов опроса может повысить общую пропускную способность системы при передаче сообщений за счет некоторого продления периода ожидания обработки для каждого сообщения. Во многих системах SQL Remote оптимизация времени реакции на сообщения не является первоочередной задачей.

Запрос на повторную передачу

Если при выполнении Message Agent опроса входящих сообщений одного сообщения в последовательности не хватает, тогда Message Agent не выполняет немедленного запроса на повторную передачу этого сообщения. Вместо этого устанавливается **время ожидания** Message Agent по умолчанию на один опрос.

Если номер следующего ожидаемого сообщения - 6, а обнаружено сообщение за номером 7, тогда Message Agent не предпринимает никаких действий до следующего опроса. Затем, если для данного пользователя не найдено новых сообщений, Message Agent направляет запрос на повторную передачу.

Можно изменить количество опросов, на время которых Message Agent делает паузу перед отправкой запроса, с помощью параметра -гр. Этот параметр часто используется вместе с параметром -rd, задающим период опроса.

Например, если период опроса слишком краток, а система передачи сообщений не сохраняет порядка поступления сообщений, тогда часто возникают ситуации, при которых рассинхронизированные сообщения поступают только по завершении двух или трех опросов. В этом случае необходимо установить для Message Agent более продолжительное время ожидания перед отправкой запроса на повторную передачу путем увеличения значения -гр. В противном случае существует вероятность отправки большого количества ненужных запросов на повторную передачу.

Пример

Предположим, что существуют два удаленных пользователя с именами **user1** и **user2**, и введена следующая командная строка Message Agent:

```
dbremote -rd 30s -rp 3
```

В следующей последовательности операций сообщения отмечены как *userX.n*, т.е., например, сообщение **user1.5** является шестым сообщением после сообщения **user1**. Message Agent ожидает, что сообщения начинаются с номера 1 для обоих пользователей.

При времени 0 секунд:

- 1 Message Agent считывает сообщения user1.1, user2.4
- 2 Message Agent обрабатывает сообщение user1.1
- 3 Теперь время ожидания Message Agent - user1: нет, user2: 3, поскольку сообщение, выходящее за пределы последовательности, поступило от user2.

При времени 30 секунд:

- 1 При чтении Message Agent обнаруживает, что новых сообщений нет.
- 2 Message Agent не производит обработку каких-либо сообщений.
- 3 Теперь время ожидания Message Agent - user1: нет, user2: 2

При времени 60 секунд:

- 1 Message Agent считывает сообщение user1.3
- 2 Message Agent не производит обработку каких-либо новых сообщений.
- 3 Время ожидания Message Agent - user1: 3, user2: 1

При времени 90 секунд:

- 1 Message Agent считывает сообщение user1.4.
- 2 Message Agent не производит обработку каких-либо сообщений.
- 3 Время ожидания Message Agent - user1: 3, user2: 0
- 4 Message Agent отправляет запрос на повторную передачу сообщения от user2.

При получении пользователем нового сообщения происходит сброс счетчика времени ожидания Message Agent, даже если это сообщение не является ожидаемым сообщением.

Настройка процесса отправки сообщений

Задержка при репликации регулируется периодичностью отправки сообщений с каждого узла и периодичностью выполнения опросов на наличие входящих сообщений. Для установки короткой задержки по времени между вводом данных и их репликацией можно задать небольшое значение параметра `-sd` Message Agent, управляющего периодичностью опросов, для проверки возможной необходимости отправки большого объема данных.

Положения, которые необходимо учитывать

Положения, которые необходимо принимать во внимание во время настройки процесса отправки сообщений, схожи с аспектами настройки периодичности опросов на наличие входящих сообщений:

- ◆ **Обычные сообщения.** Пользователь может настроить периодичность отправки обновлений в удаленные базы данных.
- ◆ **Запросы на повторную передачу.** При запросе удаленного пользователя о повторной передаче сообщения Message Agent должен выполнить определенное действие, которое прервет отправку обычных сообщений. Пользователь может контролировать степень срочности, с которой должны обрабатываться запросы о повторной передаче.
- ◆ **Количество и размер сообщений.** Если сообщения отправляются очень часто, тогда наиболее высокую вероятность отправки имеют наиболее короткие сообщения. Если сообщения отправляются реже, это обеспечивает возможность группирования в одном сообщении большого количества команд. Если при передаче большого количества маленьких сообщений в системе передачи сообщений могут возникнуть проблемы, то, возможно, придется отказаться от использования слишком коротких периодов опроса.

☞ Для получения дополнительной информации о настройке опроса для входящих сообщений см. раздел "Настройка опроса входящих сообщений" на стр. 198.

Период опроса

Управление периодом времени между опросами для отправки большого объема данных из журнала транзакций осуществляется при помощи параметра `-sd`; значение этого периода по умолчанию – 1 минута. В следующем примере период опроса устанавливается на 30 секунд:

```
dbremote -sd 30s ...
```

Точно так же опрос можно выполнять реже (см. следующую командную строку), например, каждые 5 минут:

```
dbremote -sd 5
```

Указание слишком короткого периода может оказать неблагоприятное воздействие на общую производительность системы по следующим причинам:

- ◆ Результатом слишком коротких интервалов между опросами становится большое количество коротких сообщений. Если такое количество сообщений ведет к перегрузке системы передачи сообщений, то это может повлиять на производительность.

Установка более продолжительных периодов опроса может повысить общую пропускную способность системы при передаче сообщений за счет некоторого продления периода ожидания обработки для каждого сообщения. Во многих системах SQL Remote оптимизация времени реакции на сообщения не является первоочередной задачей.

Повторная передача сообщений

При запросе пользователя о повторной передаче сообщения последнее необходимо извлечь из наиболее ранних записей журнала транзакций. Возврат к журналу транзакций для извлечения этого сообщения и его отправки прервет стандартный процесс отправки сообщений Message Agent. При поиске оптимальной производительности системы SQL Remote необходимо найти верное соотношение между срочностью отправки запросов на повторную передачу сообщений и обработкой обычных сообщений.

Параметр `-ru` управляет степенью срочности запросов на повторную передачу сообщений. Значение этого параметра указывается в минутах (либо в других единицах измерения времени, если в конце значения добавлены буквы “s” – секунды - или “h” - час). Значение по умолчанию - 0.

Для задержки обработки запросов на повторную передачу сообщений в Message Agent до поступления большего количества сообщений и запрета прерывания процесса отправки обычных сообщений следует установить в этом параметре более продолжительное время.

Следующая командная строка задает время задержки начала обработки запроса на повторную передачу сообщения в один час.

```
dbremote -ru 1h ...
```

Если параметр `-ru` не задан, тогда Message Agent выбирает значение по умолчанию на основании периода отправки сообщений пользователей, которые направили запрос на повторную передачу данных. Интервал времени между получением запроса на повторную передачу сообщения пользователю и повторное сканирование журнала не превышает половины периода отправки сообщений для данного пользователя.

Кодирование и сжатие сообщений

Сообщения поступают через системы электронной почты и другие системы передачи сообщений, поэтому не исключена опасность того, что они могут быть повреждены. Например, некоторые системы передачи сообщений используют определенные символы или комбинации символов в качестве управляющих.

Размер сообщения влияет на скорость его передачи в системе. Сжатые сообщения обрабатываются системой более быстро, нежели распакованные. С другой стороны, сам процесс сжатия может занять достаточно много времени.

Кодирование и сжатие SQL Remote

В SQL Remote имеется схема кодирования и сжатия сообщений, интегрированная в Message Agent. Данная схема имеет следующие особенности:

- ◆ **Совместимость.** Систему можно настроить на совместимость с более ранними версиями программного обеспечения.
- ◆ **Сжатие.** Пользователь может выбрать степень сжатия сообщений.
- ◆ **Кодирование.** SQL Remote кодирует сообщения с целью их прохождения через системы передачи сообщений неповрежденными. Схема кодирования и доступность дополнительных функций устанавливаются в соответствии с потребностями конкретного пользователя.

Установка параметров для обеспечения совместимости

Для обеспечения совместимости с более ранними версиями программного обеспечения необходимо установить значение -1 (минус один) параметра COMPRESSION в каждой базе данных, имеющей ПО версии 6. Эта установка обеспечивает отправку сообщений в формате, совместимом с более ранними версиями ПО.

Обновление SQL Remote

При первом обновлении Message Agent консолидированной базы данных необходимо установить его параметр базы данных COMPRESSION в -1. Поскольку каждый удаленный узел системы репликации обновляется под версию 6, то для параметра COMPRESSION можно установить значение в диапазоне от 0 (отсутствие сжатия) до 9 (максимальное сжатие). Это позволяет использовать функции сжатия сообщений при их отправке в консолидированную базу данных. После обновления всех удаленных узлов для параметра COMPRESSION Message Agent консолидированного узла можно установить значение, отличное от -1.

Кроме этого, установка параметра COMPRESSION в значение, отличное от -1, позволяет воспользоваться усовершенствованными функциями кодирования версии 6.

Схема кодирования

По умолчанию используется следующая схема кодирования сообщений SQL Remote:

- ◆ Для систем передачи сообщений, поддерживающих двоичные форматы сообщений, кодирование не выполняется.
- ◆ Для некоторых систем передачи сообщений, включая SMTP, VIM и MAPI, требуется использование текстовых форматов сообщений. Для этих систем DLL кодирования (*dbencod.dll* для Adaptive Server Anywhere и *ssencod.dll* для Adaptive Server Enterprise) преобразовывает сообщения в текстовый формат перед их отправкой. Формат сообщений на стороне приема, использующей ту же DLL, не кодируется.
- ◆ В SQL Remote также можно использовать специальную схему кодирования. Средства формирования специальной схемы кодирования описаны в следующем разделе.

- ◆ Если параметр COMPRESSION базы данных установлен в -1, для всех систем передачи сообщений используется схема кодирования, совместимая с версией 5.

Создание специальной схемы кодирования

Специальную схему кодирования можно реализовать путем создания пользовательской DLL кодирования. Эту DLL можно использовать для применения специальных функций, необходимых для особых систем передачи сообщений, либо для сбора статистики, например, количества сообщений или байт, отправленных каждому пользователю.

Заголовочный файл *dbrmt.h*, установленный в подпапке *h* папки установки, обеспечивает прикладной программный интерфейс для формирования такой схемы.

Для подачи команды SQL Remote на использование новой созданной DLL при работе с определенной системой передачи сообщений для этой системы необходимо сделать запись в реестре. Запись в реестре должна быть сделана в следующем местоположении:

```
Software
  \Sybase
    \SQL Remote
      \система-передачи-сообщений
        \encode_dll,
```

где *система-передачи-сообщений* - одна из систем передачи сообщений SQL Remote (**file**, **smtp** и т.д.). В этой записи в реестре должно содержаться имя созданной DLL кодирования.

Совместимость схем кодирования и декодирования

При реализации специальной схемы кодирования для корректного декодирования сообщений необходимо убедиться в наличии соответствующей DLL на стороне приема, а также в том, что DLL установлена в нужное местоположение.

Система отслеживания сообщений

В SQL Remote имеется система отслеживания сообщений, позволяющая контролировать обработку всех операций репликации в надлежащем порядке, не пропускать их и не выполнять одну и ту же операцию дважды.

Сбои системы передачи сообщений могут привести к тому, что сообщения репликации не будут доходить до адресата (либо будут доходить в поврежденном виде). Кроме того, сообщения могут доходить до адресата не в том порядке, в каком они были отправлены. В данном разделе описывается функция SQL Remote, предназначенная для обнаружения и исправления ошибок системы передачи сообщений, а также для обеспечения корректной обработки сообщений.

При использовании электронной почты в случае, если имеют место сбои при отправке и получении сообщений SQL Remote, необходимо убедиться в корректной работе почтовой системы между двумя компьютерами.

Работа системы отслеживания сообщений SQL Remote основана на информации о статусе, поддерживаемой в системной таблице **remoteuser** SQL Remote. Данная таблица поддерживается из Message Agent. Message Agent базы данных подписчика отправляет подтверждение в базу данных издателя для обеспечения корректного ведения таблицы **remoteuser** на обеих сторонах, участвующих в подписке.

Для Adaptive Server Anywhere функции таблицы **remoteuser** выполняет системная таблица **sys.sysremoteuser**. Для Adaptive Server Enterprise это таблица **sr_remoteuser**.

Информация о статусе в таблице remoteuser

Системная таблица **remoteuser** SQL Remote содержит строку для каждого подписчика с информацией о статусе для сообщений, отправленных и полученных этим подписчиком. В консолидированной базе данных **remoteuser** содержит строку для каждого удаленного пользователя. В каждой удаленной базе данных **remoteuser** содержит одну строку, содержащую информацию для консолидированной базы данных. (Помните о том, что подписка на публикации консолидированной базы данных осуществляется из удаленной базы данных.)

На каждой стороне подписки за ведение системной таблицы SQL Remote **remoteuser** отвечает Message Agent.

Отслеживание сообщений с помощью смещений в журнале транзакций

Информация, полученная в результате отслеживания сообщений, хранится в виде смещений в журналах транзакций баз данных подписчика и издателя. Каждый оператор COMMIT отмечен в журнале транзакций четким смещением. Порядок транзакций можно определить сравнением значений их смещений.

Упорядочение сообщений

При отправке сообщения располагаются в порядке смещения последнего оператора COMMIT предыдущего сообщения. Если в транзакцию входят несколько сообщений, то в рамках транзакции имеется идентификационный номер для правильного расположения сообщений. Максимальный размер сообщений по умолчанию - 50 000 байт, но эту настройку можно изменить при помощи параметра -I Message Agent.

Отправка сообщений

В столбце **log_sent** содержится смещение в локальном журнале транзакций для последнего сообщения, отправленного подписчику. При отправке сообщения

Message Agent он устанавливает значение **log_sent** в значение смещения последнего оператора COMMIT в сообщении.

После получения и обработки сообщения базе данных подписчика издателю немедленно отправляется подтверждение. При получении этого подтверждения Message Agent издателя устанавливает для столбца **confirm_sent** этого подписчика значение смещения из локального журнала транзакций. Значения **log_sent** и **confirm_sent** являются значениями смещений в журнале транзакций локальной базы данных, причем **confirm_sent** не может быть более поздним смещением, чем **log_sent**.

Получение сообщений

При получении и обработке обновлений репликации Message Agent в базе данных подписчика обновляет столбец **log_received**, устанавливая значение смещения последнего оператора COMMIT в сообщении. Поэтому столбец **log_received** в любой базе данных подписчика содержит значение смещения журнала транзакций в журнале транзакций базы данных издателя. После получения и обработки инструкций Message Agent отправляет подтверждение в базу данных издателя, а также устанавливает значение **confirm_received** в локальной таблице SYSREMOTEUSER. Столбец **confirm_received** в любой базе данных подписчика содержит значение смещения журнала транзакций в журнале транзакций базы данных издателя.

Двунаправленность подписок

Подписки SQL Remote являются двунаправленными: каждая удаленная база данных является подписчиком на публикации консолидированной базы данных, а консолидированная база данных подписывается на соответствующую публикацию из каждой удаленной базы данных. Поэтому в системных таблицах **remoteuser** SQL Remote в консолидированной и удаленной базах данных содержится дополнительная информация.

Message Agent обрабатывает транзакции и атомарно обновляет значение **log_received**. Если сообщение содержит несколько транзакций и во время обработки сообщения возникает ошибка, значение **log_received** точно соответствует тому, которое было обработано и подтверждено.

Повторная передача сообщений

Таблица **remoteuser** SQL Remote содержит еще два столбца, предназначенные для управления повторной передачей сообщений. Столбцы **resend_count** и **rereceive_count** являются счетчиками попыток, значения которых увеличиваются на единицу в случаях, когда сообщения по каким-то причинам теряются или удаляются.

В общем случае, в столбце **log_send** содержится то же значение, что и в столбце **log_sent**. Однако если значение в столбце **log_send** больше значения в столбце **log_sent**, Message Agent отправляет сообщения подписчику сразу при очередном запуске.

Обработка потерянных или поврежденных сообщений

При получении сообщений в базе данных подписчика Message Agent обрабатывает их в правильном порядке (определенном по смещениям в журнале) и отправляет подтверждение издателю. Если какое-либо сообщение отсутствует, Message Agent увеличивает значение локального счетчика **rereceive_count** и посылает запрос на повторную передачу этого сообщения. Другие поступившие или поступающие сообщения не обрабатываются.

При получении запроса от подписчика на повторную передачу сообщения увеличивается значение счетчика **resend_count** в базе данных издателя, а также выполняется установка **log_sent** издателя в значение **confirm_sent**. Результатом такого изменения значения **log_sent** становится повторная передача операций.

Невозможность изменения значения log_sent вручную

Пользователь не может переустановить значение log_sent, поскольку оно содержится в системной таблице.

Идентификация сообщений

Идентификация каждого сообщения производится по трем значениям:

- ◆ **resend_count**.
- ◆ Смещение в журнале транзакций последнего оператора COMMIT в предыдущем сообщении.
- ◆ Идентификационный номер в пределах транзакции (для транзакций, включающих в себя обмен несколькими сообщениями).

Сообщения со значением **resend_count** меньшим, чем **rereceive_count**, не обрабатываются и удаляются. Таким образом обеспечивается однократное выполнение каждой операции.

Администрирование SQL Remote для Adaptive Server Anywhere

Об этой главе

В данной главе подробно рассматриваются вопросы настройки и управления системой для администраторов SQL Remote, использующих Adaptive Server Anywhere в качестве консолидированной базы данных.

Содержание

Раздел	Страница
Работа с Message Agent	210
Сообщения об ошибках и их обработка	212
Управление журналом транзакций и резервным копированием	216
Использование режима ретрансляции	225

Работа с Message Agent

В данном разделе описывается процесс запуска и дальнейшей работы с Message Agent для Adaptive Server Anywhere.

☞ Для получения информации об особенностях Message Agent, общих для Adaptive Server Anywhere и Adaptive Server Enterprise, см. раздел "Администрирование SQL Remote" на стр. 173.

Запуск Message Agent

Message Agent имеет набор параметров, управляющих его поведением. Единственный параметр, необходимый для запуска Message Agent, – это параметр подключения (-c).

☞ Для получения дополнительной информации о параметрах подключения см. раздел "Параметры подключения" (Connection parameters) на стр. 70 в документе "Руководство по администрированию баз данных ASA" (ASA Database Administration Guide).

Ключевое слово для расширенного режима	Сокращенная форма	Аргумент
DatabaseFile	DBF	строка
DatabaseName	DBN	строка
DatabaseSwitches	DBS	строка
EngineName	ENG	строка
Password	PWD	строка
Start	Start	строка
Userid	UID	строка

Работа с Message Agent в качестве службы

Если Message Agent работает в режиме непрерывной работы (не в пакетном режиме), то, возможно, потребуется обеспечить постоянную работу Message Agent во время работы сервера.

Это можно сделать путем запуска Message Agent как **службы** Windows. Можно настроить конфигурацию службы так, что она будет работать даже после выхода текущего пользователя из системы и запускаться при очередном запуске операционной системы.

☞ Полное описание работы программ в качестве служб см. в разделе "Работа с сервером базы данных" (Running the Database Server) на стр. 3 в документе "Руководство по администрированию баз данных ASA" (ASA Database Administration Guide).

Message Agent и безопасность репликации

В учебных разделах предыдущей главы Message Agent запускался с помощью идентификатора пользователя с полномочиями администратора БД. Операции в сообщениях выполняются по идентификатору пользователя, указанному в строке подключения Message Agent; использование идентификатора пользователя **DBA** гарантирует наличие у данного пользователя всех необходимых полномочий для внесения изменений в конфигурацию системы.

Во многих ситуациях сообщение идентификатора пользователя и пароля администратора БД всем пользователям удаленной базы данных неприемлемо из соображений безопасности и конфиденциальности данных. В SQL Remote предусмотрено решение, обеспечивающее Message Agent полный доступ к базе данных для внесения любых изменений в сообщения без нарушения системы безопасности.

Особые полномочия REMOTE DBA имеют следующие свойства:

- ◆ **Отсутствие полномочий при отключении от Message Agent.** Пользователь, которому предоставлены полномочия REMOTE DBA, не имеет дополнительных привилегий на какое-либо подключение, кроме Message Agent. Поэтому даже при большом количестве пользователей, знающих идентификатор и пароль пользователя REMOTE DBA, проблемы нарушения безопасности не возникает. До тех пор, пока идентификатору пользователя не назначаются никакие полномочия, кроме полномочий CONNECT, предоставленные для базы данных, никто не сможет использовать этот идентификатор пользователя для доступа к данным в базе данных.
- ◆ **Полные полномочия администратора БД от Message Agent.** При подключении с Message Agent пользователь с полномочиями REMOTE DBA имеет полные полномочия администратора БД на доступ к базе данных.

Использование
полномочий
REMOTE DBA

Рекомендуется предоставить издателю и каждому удаленному пользователю полномочия REMOTE DBA на работу с консолидированной базой данных. При извлечении удаленной базы данных удаленный пользователь становится издателем удаленной базы данных, и ему предоставляются те же полномочия, что и в консолидированной базе данных, включая полномочия REMOTE DBA, позволяющее применять данный идентификатор пользователя в строке подключения Message Agent. Принятие этой процедуры означает отсутствие каких-либо дополнительных идентификаторов пользователей для администрирования, а также то, что каждому удаленному пользователю необходимо знать только один идентификатор пользователя для доступа к базе данных - из Message Agent (который, в этом случае, имеет полные полномочия администратора БД) или из любого другого клиентского приложения (в этом случае полномочия REMOTE DBA не влекут за собой предоставление дополнительных полномочий).

Предоставление
полномочий
REMOTE DBA

Полномочия REMOTE DBA можно предоставить пользователю с именем **dbremote** следующим образом:

```
GRANT REMOTE DBA
TO dbremote
IDENTIFIED BY dbremote
```

В Adaptive Server Anywhere полномочия REMOTE DBA можно предоставить удаленному пользователю путем установки соответствующего параметра на закладке Authorities в окне свойств удаленного пользователя.

Сообщения об ошибках и их обработка

В данном разделе описывается процесс выявления и обработки ошибок в Message Agent.

Обработка ошибок по умолчанию

По умолчанию при возникновении ошибки Message Agent добавляет соответствующую запись в выводимой информации журнала. Message Agent отправляет выводимую информацию с этой записью в окно для просмотра пользователем или файл журнала. По умолчанию эта информация отображается только в окне; при использовании параметра `-o` также осуществляется вывод в файл журнала.

В выводимом файле журнала может содержаться больше информации, чем в окне. В файл журнала Message Agent входит следующее:

- ◆ Список обработанных сообщений;
- ◆ Список невыполненных операторов SQL;
- ◆ Список прочих ошибок.

Конфликты UPDATE не являются ошибками

Конфликты UPDATE не являются ошибками, поэтому отчеты о них не направляются в Message Agent.

☞ Для получения дополнительной информации о файле журнала см. раздел "Message Agent" на стр. 9.

Игнорирование ошибок

Бывают случаи, когда ошибку, выявленную Message Agent во время обработки операторов SQL, требуется оставить без внимания. Это может произойти тогда, когда известны условия, при которых возникла ошибка, и пользователь уверен в том, что ее результатом не станет несогласованность данных, а последствия этой ошибки не будут существенными.

Для отключения функции сообщения об ошибках для действия, вызывающего известную ошибку, можно создать триггер BEFORE. Этот триггер должен сообщить значение REMOTE_STATEMENT_FAILED SQLSTATE (5RW09) или SQLCODE (-288).

Например, при необходимости отмены выполнения операторов INSERT в таблице, не обрабатываемой из-за отсутствия столбца, на который указывает ссылка, можно создать триггер BEFORE INSERT, который запускает REMOTE_STATEMENT_FAILED SQLSTATE в случае, если столбца, на который указывает ссылка, не существует. Оператор INSERT не выполняется, но эта ошибка не заносится в журнал Message Agent.

Применение процедур обработки ошибок

Помимо занесения в журнал сообщений об ошибках, SQL Remote позволяет выполнять некоторые другие процессы. Параметр Replication_error базы данных позволяет задать хранимую процедуру, вызываемую Message Agent при возникновении ошибок. По умолчанию никакие процедуры не вызываются.

В процедуре должен быть один аргумент типа CHAR, VARCHAR или LONG VARCHAR. Эта процедура вызывается дважды: один раз с сообщением об ошибке, и один раз – с оператором SQL, вызывающим ошибку.

Несмотря на то, что данный параметр позволяет отслеживать ошибки и контролировать их появление при репликации, возможность возникновения ошибок необходимо минимизировать на этапе настройки системы, поскольку данный параметр не обеспечивает устранения таких ошибок.

Например, данная процедура могла привести к вставке ошибки в таблицу с текущим временем и идентификатором удаленного пользователя, после чего эта информация была бы реплицирована в консолидированную базу данных. Приложение в консолидированной базе данных при обнаружении ошибок сгенерирует сообщение об ошибке или отправит администратору электронное письмо.

☞ Для получения информации о параметре REPLICATION_ERROR см. раздел "Параметры SQL Remote" на стр. 272.

Пример: отправка уведомлений об ошибках по электронной почте

Возможно, что потребуется получать по электронной почте уведомления об ошибках, которые регистрирует Message Agent. В данном разделе описывается способ отправки администратору электронных сообщений при возникновении ошибок.

Хранимая процедура

Хранимая процедура для этого примера называется **sp_LogReplicationError** и принадлежит пользователю **cons**. Для вызова этой процедуры при появлении ошибки задайте параметр **Replication_error** базы данных из Interactive SQL или Sybase Central:

```
SET OPTION PUBLIC.Replication_error =
    'cons.sp_LogReplicationError'
```

Это уведомление реализуется следующей хранимой процедурой:

```
CREATE PROCEDURE cons.sp_LogReplicationError
    (IN error_text LONG VARCHAR)
BEGIN
    DECLARE current_remote_user CHAR(255);
    SET current_remote_user = CURRENT REMOTE USER;

    // Регистрация ошибки
    INSERT INTO cons.replication_audit
        ( remoteuser, errormsg)
    VALUES
        ( current_remote_user, error_text);
    COMMIT WORK;

    //Уведомление администратора БД об ошибке
    // по электронной почте. Данная информация
    // необходима при появлении ошибки в
консолидированной
    // базе данных.
    // В электронном сообщении должны быть приведены
    // строки с ошибкой.
    // Message Agent переходит к выполнению процедуры
    IF CURRENT PUBLISHER = 'cons' THEN
        CALL sp_notify_DBA( error_text );
    END IF
END;
```

Данная хранимая процедура вызывает другую хранимую процедуру, управляющую отправкой электронного сообщения:

```
CREATE PROCEDURE sp_notify_DBA(in msg long varchar)
BEGIN
    DECLARE rc INTEGER;
```

```

rc=call xp_startmail(mail_user='davidf');
//При успешном получении доступа к почте
IF rc=0 THEN
rc=call xp_sendmail(
    recipient='Doe, John; John, Elton',
    subject='SQL Remote Error',
    "message"=msg);
//Если почта успешно отправлена, остановка
IF rc=0 THEN
    call xp_stopmail()
    END IF
    END IF
END IF
END;

```

Таблица ревизий

Таблицу ревизий можно определить следующим образом:

```

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)
);

```

Ниже приведено описание столбцов в этой таблице:

Столбец	Описание
pub	Текущий издатель базы данных (сообщает о том, в какую базу данных добавлена запись).
remoteuser	Удаленный пользователь, обрабатывающий сообщение (идентифицирует базу данных-источник).
errmsg	Сообщение об ошибке, переданное в процедуру Replication_error.

Ниже приведена типовая вставка в таблицу из ошибки, приведенной выше:

```

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

```

Поскольку Adaptive Server Anywhere поддерживает вызов внешних DLL из хранимых процедур, то вместо использования электронной почты можно спроектировать систему пейджинговой связи.

Пример ошибки

Например, если строка вставлена в консолидированную базу данных при помощи первичного ключа (так же, как и в удаленную базу данных), тогда Message Agent отобразит следующие ошибки:

```

Received message from "cons" (0-0000000000-0)
SQL statement failed: (-193) primary key for table 'reptable' is not unique
INSERT INTO cons.reptable(id,text,last_contact) VALUES
(2,'dave','1997/apr/21 16:02:38.325') COMMIT WORK

```

Среди сообщений, поступивших на имя пользователей Доу, Джона и Элтона, каждое сообщение Джону содержит ошибку SQL Remote:

```
primary key for table 'reptable' is not unique (-193)
```

```
INSERT INTO cons.reptable(id,text,last_contact) VALUES  
(2,'dave','1997/apr/21 16:02:52.605')
```

Управление журналом транзакций и резервным копированием

Важность
регулярного
создания резервных
копий

Успешное выполнение репликации зависит от наличия доступа к операциям в журнале транзакций; также иногда необходим доступ к старым журналам транзакций. В данном разделе описана установка процедур резервного копирования в консолидированной и удаленных базах данных для корректного доступа к старым журналам транзакций.

В узлах консолидированной базы данных SQL Remote очень важно применять практику регулярного создания резервных копий. Утеря журнала транзакций может привести к необходимости повторного извлечения удаленных баз данных. В узле консолидированной базы данных рекомендуется создавать зеркальную копию журнала транзакций.

☞ Для получения информации о зеркальных копиях журнала транзакций и других процедурах резервного копирования см. раздел "Резервное копирование и восстановление данных" (Backup and Data Recovery) на стр. 299 в документе *"Руководство по администрированию баз данных ASA" (ASA Database Administration Guide)*.

Обеспечение
доступа к старым
транзакциям

Все журналы транзакций должны быть гарантированно доступными для системы репликации до тех пор, пока в них не отпадет необходимость.

Во многих установках пользователи удаленных баз данных могут получать обновления с главного сервера (при необходимости – ежедневно). Если какие-то сообщения утеряны или удалены, но их необходимо отправить повторно через систему отслеживания, то возможно, что в них потребуется включить изменения, внесенные несколько дней назад. Если во время отпуска удаленного пользователя какие-то сообщения оказываются утерянными, тогда могут потребоваться изменения, сделанные несколько недель назад. Если резервные копии журнала транзакций создаются ежедневно, тогда журнал с изменениями не будет запускаться на сервере.

Из-за того, что журнал транзакций непрерывно увеличивается в размере, может возникнуть проблема наличия свободного места на диске. Для регулирования размера журнала транзакций можно использовать обработчик событий, с помощью которого будет осуществляться переименование журнала по достижении им заданного размера. После этого можно использовать параметр DELETE_OLD_LOGS для удаления ненужных файлов из журнала.

☞ Для получения дополнительной информации об управлении размером журнала транзакций см. раздел "Оператор BACKUP" (BACKUP statement) на стр. 245 в документе *"Справочник по SQL для ASA" (ASA SQL Reference Manual)*.

Установка папки журнала транзакций

Когда Message Agent требуется отсканировать журналы транзакций, исключая текущий, он просматривает все файлы журнала, находящиеся в указанной **папке журнала транзакций**. Нужная папка указывается для Message Agent в командной строке.

Пример

Например, следующая командная строка подсказывает Message Agent обратиться к папке `e:\archive` для поиска старых журналов транзакций. Эту команду необходимо ввести одной строкой.

```
dbremote -c "eng=имя_сервера;uid=DBA;pwd=SQL" e:\archive
```


Имена журналов не имеют значения

Message Agent открывает все файлы в папке журнала транзакций для определения того, какие файлы являются журналами, поэтому фактические имена файлов журналов транзакций значения не имеют.

В данном разделе описан процесс установки процедуры резервного копирования для поддержания папки журнала транзакций.

Параметры утилиты резервного копирования

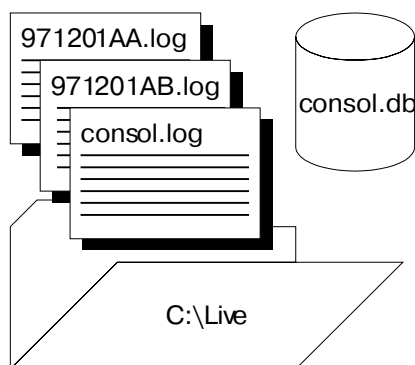
Утилита резервного копирования Adaptive Server Anywhere имеет несколько управляющих ее поведением параметров, доступных в соответствующем мастере Sybase Central или в качестве параметра *dbbackup*.

Здесь описаны два подхода к применению утилиты резервного копирования при копировании консолидированной базы данных SQL Remote. Резервные копии должны обеспечивать доступность набора журналов транзакций, необходимых для использования Message Agent.

Использование текущей папки в качестве папки журнала транзакций

Данный параметр рекомендуется для использования при переименовании и повторном запуске журнала транзакций при резервном копировании журналов транзакций консолидированной и удаленной баз данных. Для утилиты *dbbackup* это параметр *-r*.

На рисунке ниже показана база данных, названная *consol.db*, с журналом транзакций с именем *consol.log* в той же папке. В данном случае для упрощения процедуры предполагается, что журнал расположен в той же папке, что и база данных, хотя в реальной ситуации такое расположение не всегда способствует безопасности. Папка имеет название *c:\live*.



Командная строка резервного копирования

Следующая командная строка создает резервную копию базы данных, используя параметры переименования и перезапуска:

```
dbbackup -r -c "uid=DBA;pwd=SQL" c:\archive
```

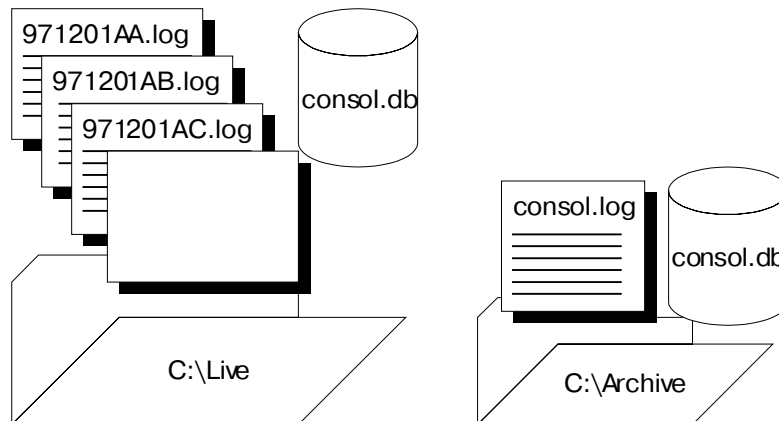
Для каждой базы данных параметры строки подключения будут различными.

Шаги при резервном копировании

При создании резервной копии журнала транзакций в папку *c:\archive* с использованием параметра переименования и перезапуска утилита резервного копирования выполняет следующие задачи:

- 1 Создание резервной копии журнала транзакций через создание резервного файла *c:\archive\consol.log*;
- 2 Переименование существующего файл журнала транзакций в *971201xx.log*, где *xx* - последовательные символы от *AA* до *ZZ*;
- 3 Запуск нового журнала транзакций *consol.log*.

После нескольких резервных копирований текущая папка будет содержать набор последовательных журналов транзакций.



Командная строка Message Agent

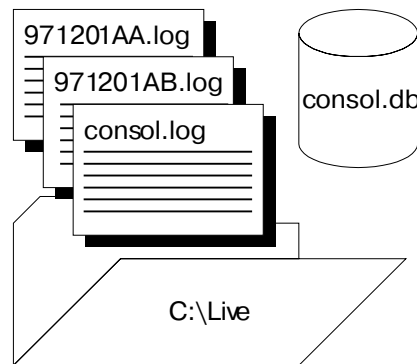
Message Agent можно запустить с доступом к этим файлам журналов, используя следующую командную строку:

```
dbremote -c "dbn=hq;..." c:\live
```

Использование папки резервного копирования в качестве папки журнала транзакций

Альтернативной процедурой является использование папки резервного копирования в папке журнала транзакций.

На рисунке ниже показана база данных, названная *consol.db*, с журналом транзакций с именем *consol.log* в той же папке. В данном случае для упрощения процедуры предполагается, что журнал расположен в той же папке, что и база данных, хотя в реальной ситуации такое расположение не всегда является безопасным. Папка имеет название *c:\live*.



Командная строка резервного копирования

Следующая командная строка создает резервную копию базы данных, используя параметр переименования и перезапуска, а также использует параметр для переименования резервного файла журнала транзакций:

```
dbbackup -r -k -c "uid=DBA;pwd=SQL" c:\archive
```

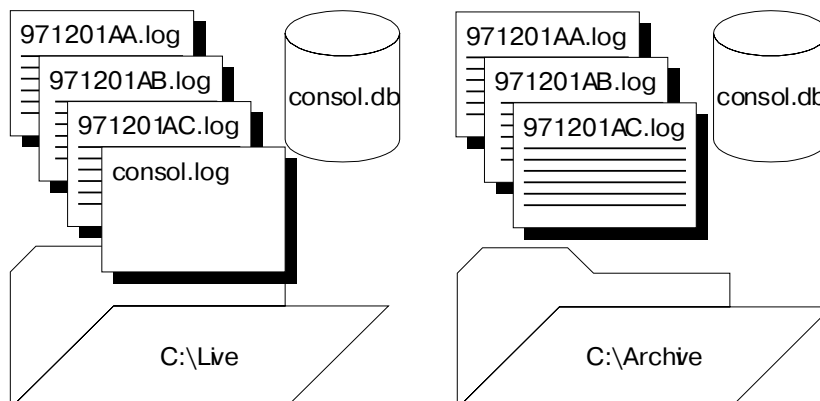
Для каждой базы данных параметры строки подключения будут различными.

Шаги при резервном копировании

При создании резервной копии журнала транзакций в папке *c:\archive* с использованием параметра переименования и перезапуска и параметра переименования журнала транзакций утилита резервного копирования выполняет следующие задачи:

- 1 Переименование существующего файла журнала транзакций в *971201xx.log*, где *xx* - последовательные символы от *AA* до *ZZ*;
- 2 Создание резервной копии файла (*971201xx.log*) журнала транзакций в папке резервного копирования;
- 3 Запуск нового журнала транзакций *consol.log*.

После нескольких операций резервного копирования текущая и архивная папки будут содержать набор последовательных журналов транзакций.



Командная строка
Message Agent

Message Agent можно запустить с доступом к этим файлам журналов, используя следующую командную строку:

```
dbremote -c "dbn=hq;..." c:\archive
```

Имена старых журналов до версии 8.01
До версии 8.01 Adaptive Server Anywhere старые файлы журналов транзакций назывались по шаблону *ггммдд01.log*, *ггммдд02.log* и т.д. Для обеспечения возможности хранения большего количества журналов транзакций было произведено изменение формата имени. Поскольку Message Agent просматривает все файлы в указанной папке независимо от их имен, изменение формата имени не должно повлиять на существующие приложения.

Управление старыми журналами транзакций

Все журналы транзакций должны быть гарантированно доступными до тех пор, пока у системы репликации в них не отпадет необходимость: в этом случае их можно удалить.

Система репликации не требует наличия старых журналов транзакций после того, как все удаленные базы данных получили и успешно обработали сообщения, содержащиеся в файлах журналов. Удаленные базы данных подтверждают успешное получение сообщений из консолидированной базы данных, и такое подтверждение вызывает установку соответствующих значения в таблицах SQL Remote консолидированной базы данных (см. раздел "Система отслеживания сообщений" на стр. 205). Старые журналы транзакций в консолидированной базе данных больше не требуются SQL Remote, если это подтверждение уже получено из всех удаленных баз данных.

Использование
параметра
Delete_old_logs

Для автоматического управления старыми журналами транзакций в консолидированной базе данных можно использовать параметр `Delete_old_logs`.

По умолчанию параметр `DELETE_OLD_LOGS` базы данных установлен в `OFF`. Если он установлен в `ON`, тогда Message Agent автоматически удаляет старые журналы транзакций, когда в них уже нет необходимости. Журнал транзакций становится ненужным, когда все подписчики подтвердили получение всех изменений, записанных в этом журнале.

Задать параметр `DELETE_OLD_LOGS` можно либо для группы `PUBLIC`, либо только для пользователя, содержащегося в строке подключения Message Agent.

Пример

```
◆ Следующий оператор задает DELETE_OLD_LOGS:  
SET OPTION PUBLIC.DELETE_OLD_LOGS = 'ON'
```

Восстановление при отказе носителя базы данных (консолидированная база данных)

В данном разделе описан процесс восстановления при отказе устройства, на котором хранится консолидированная база данных.

Эта процедура выполняется наиболее просто при наличии только одного файла журнала транзакций. Несмотря на то, что эта процедура редко используется в отношении консолидированной базы данных, она описывается первой, после чего следует более распространенная, но и более сложная процедура с набором файлов журнала транзакций.

Восстановление при наличии одного журнала транзакций

В этом случае предполагается, что существует один файл журнала транзакций, который ведется с момента создания базы данных. Также предполагается, что ранее были сделаны резервные копии файла базы данных (например, на пленке), которые являются доступными.

❖ Процедура восстановления базы данных

- 1 Создайте копию базы данных и файла журнала.
- 2 Восстановите файл базы данных (.db), а не файл журнала, с пленки во временную папку.
- 3 Запустите базу данных, используя существующий журнал транзакций и параметр -a, для обработки транзакций и обновления файла базы данных.
- 4 Запустите базу данных обычным способом. Записи о любых новых действиях будут добавляться в конец журнала транзакций.



Пример

В данном примере описан процесс восстановления с использованием зеркальной копии журнала транзакций.

Предположим, что имеется файл консолидированной базы данных с именем *consol.db* в папке *c:\dbdir* и файл журнала транзакций *c:\logdir\consol.log* с зеркальной копией на *d:\mirdir\consol.mlg*.

❖ **Процедура восстановления при отказе носителя диска C:**

- 1 Создайте резервную копию зеркального журнала транзакций *d:\mirdir\consol.mlg*.
- 2 Замените неисправные аппаратные средства и повторно установите все поврежденное программное обеспечение.
- 3 Создайте временную папку, в которой будет выполняться восстановление (например, *c:\recover*)
- 4 Восстановите самую последнюю резервную копию файла базы данных *consol.db* на *c:\recover\consol.db*.
- 5 Скопируйте зеркальный журнал транзакций, *d:\mirdir\consol.mlg* в папку восстановления с расширением *.log* - *c:\recover\consol.log*.
- 6 Запустите базу данных, используя следующую командную строку:

```
dbeng8 -a C:\RECOVER\CONSOL.DB
```
- 7 Завершите работу сервера базы данных.
- 8 Создайте резервную копию восстановленной базы данных и журнала транзакций из *c:\recover*.
- 9 Скопируйте файлы из *c:\recover* в соответствующие рабочие папки:
 - ◆ Скопируйте *c:\recover\consol.db* в *c:\dbdir\consol.db*.
 - ◆ Скопируйте *c:\recover\consol.log* в *c:\dbdir\consol.IOG* и в *d:\mirdir\consol.mlg*.
- 10 Перезапустите систему в обычном режиме.

Восстановление при наличии множественных журналов транзакций

При наличии набора журналов транзакций процедура восстановления будет иной. Предположим, что на резервном носителе (например, на пленке) хранятся резервные копии файла базы данных.

❖ **Процедура восстановления базы данных**

- 1 Создайте копию базы данных и файла журнала транзакций.
- 2 Восстановите файл базы данных (*.db*), а не файл журнала, с пленки во временную папку.
- 3 Во временной папке запустите базу данных с использованием старых журналов с параметром *-a* и названных журналов транзакций в правильном порядке.
- 4 Запустите базу данных, используя текущий журнал транзакций и параметр *-a* для обработки транзакций и обновления файла базы данных.
- 5 Запустите базу данных обычным способом. Записи о любых новых действиях будут добавляться в конец журнала транзакций.

Пример

Предположим, что имеется файл консолидированной базы данных с именем *c:\dbdir\cons.db*. Файл журнала транзакций *c:\dbdir\cons.log* имеет зеркальную копию на *d:\mirdir\cons.mlg*.

Предположим, полное резервное копирование производится еженедельно, а выборочное резервное копирование - ежедневно с помощью следующей команды:

```
dbbackup -c "uid=DBA;pwd=SQL" -r -t E:\BACKDIR
```

Эта команда создает резервную копию журнала транзакций *cons.log* в папке *e:\backdir*. Затем файл журнала транзакций переименовывается в *datexx.log*, где *date* - текущая дата, а *xx* - следующая кодовая таблица в последовательности, после чего начинается внесение записей в новый журнал транзакций. После этого создается резервная копия папки *e:\backdir* с использованием других утилит.

В этом сценарии Message Agent запускается с указанием дополнительной папки для индикации переименованных файлов журнала транзакций. Командная строка Message Agent будет следующей:

```
dbremote -c "uid=DBA;pwd=SQL" C:\DBDIR
```

На третий день после еженедельного резервного копирования файл базы данных оказывается поврежденным из-за повреждений секторов диска.

❖ **Процедура восстановления отказа носителя диска C:**

- 1 Создайте резервную копию зеркального журнала транзакций *d:\mirdir\cons.mlg*.
- 2 Создайте временную папку, в которой будет выполняться восстановление. Здесь она называется *c:\recover*.
- 3 Восстановите самую последнюю резервную копию файла базы данных на *c:\recover\cons.db*.
- 4 Выполните обработку переименованных журналов транзакций в следующем порядке:

```
dbeng8 -a C:\DBDIR\date00.LOG C:\RECOVER\CONS.DB
```

```
dbeng8 -a C:\DBDIR\date01.LOG C:\RECOVER\CONS.DB
```

- 5 Скопируйте текущий журнал транзакций *c:\dbdir\cons.log* в папку восстановления – предположим, это папка *c:\recover\cons.log*.
- 6 Запустите базу данных, используя следующую команду:

```
dbeng8 C:\RECOVER\CONS.DB
```
- 7 Завершите работу сервера базы данных.
- 8 Создайте резервную копию восстановленной базы данных и журнала транзакций из *c:\recover*.
- 9 Скопируйте файлы из *c:\recover* в соответствующие рабочие папки.
 - ◆ Скопируйте *c:\recover\cons.db* в *c:\dbdir\cons.db*.
 - ◆ Скопируйте *c:\recover\cons.log* в *c:\dbdir\cons.log* и в *d:\mirdir\cons.mlg*.
- 10 Перезапустите систему в обычном режиме.

Процедуры резервного копирования в удаленных базах данных

Процедуры резервного копирования в удаленных базах данных не являются столь же важными, как в консолидированной базе данных. В качестве способа резервного копирования можно воспользоваться репликацией из консолидированной базы данных. В случае отказа носителя удаленную базу данных придется повторно извлечь из консолидированной базы данных, и все операции, для которых не была выполнена репликация, будут утеряны. (Для восстановления утерянных операций можно попробовать воспользоваться утилитой журнала транзакций).

Даже если для безопасности базы данных пользователь полагается на репликацию, все равно необходимо периодически создавать резервные копии в

удаленных базах данных для того, чтобы объем журнала транзакций не был слишком большим. Здесь применяется тот же параметр (переименование и перезапуск журнала), что и при работе с консолидированной базой данных, который позволяет запустить Message Agent так, чтобы он имел доступ к переименованным файлам журналов. При установке параметра DELETE_OLD_LOGS в ON в удаленной базе данных Message Agent будет автоматически удалять старые файлы журналов транзакций, если необходимости в них больше нет.

Автоматическое переименование журнала транзакций

Для снятия необходимости переименования журнала транзакций на удаленном компьютере при закрытом сервере базы данных можно воспользоваться параметром -x Message Agent. Параметр -x переименовывает журнал транзакций после его сканирования на наличие исходящих сообщений.

Обновление ПО консолидированных баз данных

В данном разделе описываются процедуры обновления ПО консолидированной базы данных в среде SQL Remote. Эти рекомендации также применимы к базам данных Adaptive Server Anywhere, являющимся первичными узлами в системах Sybase Replication Server.

Установка нового программного обеспечения не всегда делает доступными новые функции. Во многих случаях для доступа к новым функциям на базах данных требуется запущенная утилита обновления (Upgrade). Утилита обновления добавляет в системный каталог любую информацию, необходимую для того, чтобы сделать доступными новые функции. При работе с утилитой обновления пользователь получает сообщение о необходимости архивирования журнала транзакций. Это необходимо потому, что утилита обновления создает новый журнал транзакций с новым форматом файла.

При использовании SQL Remote или Replication Server журнал транзакций необходимо сохранять для Message Agent и Replication Agent соответственно. После запуска утилиты обновления необходимо закрыть процессор, переименовать журнал транзакций и оставить его для удаления Message Agent. В целях резервного копирования журнал также необходимо заархивировать.

☞ Для получения информации об утилите обновления см. раздел "Утилита обновления" (The Upgrade utility) на стр. 521 в документе *"Руководство по администрированию баз данных ASA" (ASA Database Administration Guide)*.

Выгрузка и перезагрузка базы данных, задействованной в репликации

Если база данных участвует в репликации, при выгрузке и перезагрузке базы данных следует соблюдать особую осторожность.

☞ Рекомендации и инструкции по выгрузке и перезагрузке базы данных см. в разделе "Обновление формата файла базы данных" (Upgrading the database file format) на стр. 144 в документе *"Новое в SQL Anywhere Studio" (What's New in SQL Anywhere Studio)*. Инструкции в данном разделе применимы к базам данных, задействованным в репликации SQL Remote.

В данном разделе описывается ручной способ выгрузки и повторной загрузки базы данных на случай каких-то особых обстоятельств, когда применение более автоматизированной процедуры, описанной выше, становится невозможным (например, изменение схемы или каких-либо другой важных свойств базы данных).

Репликация основана на журнале транзакций. После выгрузки или перезагрузки базы данных старый журнал транзакций становится недоступным. По этой

причине для участия в репликации использование проверенных методов создания резервных копий является крайне важными.

❖ **Процедура выгрузки и перезагрузки консолидированной базы данных (вручную)**

- 1 Закройте существующую базу данных.
- 2 Выполните полное резервное копирование в режиме off-line. Для этого скопируйте базу данных и файлы журналов транзакций в надежное местоположение.
- 3 Запустите утилиту *dbtran* для отображения начального и конечного смещений текущего файла журнала транзакций базы данных. Сохраните информацию о конечном смещении для использования в дальнейшем.
- 4 Переименуйте текущий файл журнала транзакций так, чтобы он не изменялся во время процесса выгрузки, и поместите этот файл в неактивную папку.
- 5 Запустите существующую базу данных.
- 6 Выгрузите базу данных.
- 7 Закройте существующую базу данных. Необходимости в этой базе данных и любом файле журнала, созданных в данном и предыдущем шагах, больше нет.
- 8 Инициализируйте новую базу данных.
- 9 Перезагрузите данные в новую базу.
- 10 Закройте новую базу данных.
- 11 Удалите текущий файл журнала транзакций для новой базы данных.
- 12 Используйте в новой базе данных *dblog* с конечным смещением, отмеченным в шаге 3 как параметр *-z*, а также установите относительное смещение на ноль.

```
dblog -x 0 -z 137829 имя_БД.db
```
- 13 При запуске Message Agent укажите в командной строке местоположение исходной неактивной папки.

Использование режима ретрансляции

Издатель консолидированной базы данных может напрямую войти в удаленные узлы при использовании режима ретрансляции, обеспечивающего стандартным операторам SQL доступ к удаленному узлу. По умолчанию операторы режима ретрансляции также выполняются в локальной (консолидированной) базе данных, но дополнительное ключевое слово препятствует локальному выполнению операторов.

Предостережение

Всегда проверяйте операции ретрансляции на тестовой базе данных с подписанной удаленной базой данных. Никогда не запускайте непроверенные сценарии ретрансляции в действующей базе данных.

Запуск и завершение ретрансляции

Режим ретрансляции запускается и завершает свою работу с помощью оператора PASSTHROUGH. Любой оператор, введенный между начальным оператором PASSTHROUGH и оператором PASSTHROUGH STOP (прерывающим выполнение режима ретрансляции), проверяется на ошибки синтаксиса, выполняется в текущей базе данных, а также передается идентифицированному подписчику и выполняется в базе данных подписчика. Операторы между операторами запуска и завершением работы режим ретрансляции называются **сеансом ретрансляции**.

Нижеприведенный оператор начинает сеанс ретрансляции, передающий операторы по списку из двух поименованных подписчиков, без выполнения в локальной базе данных:

```
PASSTHROUGH ONLY
FOR userid_1, userid_2;
```

Направление операторов ретрансляции

Следующий оператор запускает сеанс ретрансляции, передающий операторы всем подписчикам на указанную публикацию:

```
PASSTHROUGH ONLY
FOR SUBSCRIPTION TO [владелец].имя-публикации [ ( строка
) ] ;
```

В режиме ретрансляции применяется метод накопления. В следующем примере **statement_1** отправляется к **user_1**, а **statement_2** отправляется как к **user_1**, так и к **user_2**.

```
PASSTHROUGH ONLY FOR user_1 ;
statement_1 ;
PASSTHROUGH ONLY FOR user_2 ;
statement_2 ;
```

Следующий оператор завершает сеанс ретрансляции:

```
PASSTHROUGH STOP ;
```

PASSTHROUGH STOP завершает режим ретрансляции для всех удаленных пользователей.

Порядок применения операторов ретрансляции

Операторы ретрансляции реплицируются последовательно вместе с обычными сообщениями репликации, в том порядке, в котором операторы внесены в журнал транзакций.

Ретрансляция обычно используется для отправки операторов языка определения данных. В этом случае реплицированные операторы DML используют схему *before* перед ретрансляцией и схему *after* после ретрансляции.

Примечания по использованию режима ретрансляции

- ◆ Операции ретрансляции необходимо всегда тестировать на тестовой базе данных с подписанной удаленной базой данных. Нельзя запускать непроверенные сценарии ретрансляции в действующей базе данных.
- ◆ Для объектов всегда должны указываться имена владельцев.

Операторы PASSTHROUGH не выполняются в удаленных базах данных по идентификатору пользователя. Поэтому имена объектов без указания имени владельца могут быть неправильно интерпретированы.

Применение режима ретрансляции и ограничения

Режим ретрансляции – это мощный инструмент, при работе с которым следует соблюдать осторожность. Некоторые операторы, особенно операторы определения данных, могут вызвать сбой инсталляции SQL Remote. В системе SQL Remote предполагается, что каждая база данных имеет одинаковый набор объектов; если в каких-то узлах таблица изменена, а на других – нет, тогда репликация изменения данных выполнена не будет.

Кроме того, важно помнить, что в установке по умолчанию при режиме ретрансляции также выполняются операторы в локальной базе данных. Для отправления операторов в удаленную базу данных без их локального выполнения необходимо ввести ключевое слово ONLY. Следующий набор операторов удаляет таблицу не только в удаленной, но и в консолидированной базе данных.

```
-- Удаление таблицы в удаленной и локальной базах данных
PASSTHROUGH TO Joe_Remote ;
DROP TABLE CrucialData ;
PASSTHROUGH STOP ;
```

Синтаксис при удалении таблицы в удаленной базе данных следующий:

```
-- Удаление таблицы только в удаленной базе данных
PASSTHROUGH ONLY TO Joe_Remote ;
DROP TABLE CrucialData ;
PASSTHROUGH STOP ;
```

Ниже представлены задачи, которые могут быть выполнены при запуске системы SQL Remote:

- ◆ Добавление новых пользователей;
- ◆ Повторная синхронизация пользователей;
- ◆ Удаление пользователей из системы;
- ◆ Изменение адреса, типа сообщений или периодичности для удаленного пользователя;
- ◆ Добавление столбца в таблицу.

Многие другие изменения схемы, скорее всего, вызовут серьезные проблемы при их выполнении в работающей системе SQL Remote.

Ретрансляция осуществляется только на одном уровне иерархии

В многоуровневой системе SQL Remote важно, чтобы операторы ретрансляции работали на уровне баз данных, находящихся непосредственно под текущим уровнем. В многоуровневой системе операторы ретрансляции необходимо вводить в каждую консолидированную базу данных для уровня, находящегося непосредственно под ней.

Операции, не реплицируемые в режиме ретрансляции

Некоторые операторы в режиме ретрансляции стоит рассмотреть особо.

Вызов процедур

При вызове хранимой процедуры в режиме ретрансляции с использованием оператора CALL или EXEC реплицируется сам оператор CALL, но не реплицируется ни один из операторов в рамках процедуры. Предполагается, что процедура на стороне репликации является допустимой и выполняется правильно.

Управление потоками операторов и операциями курсора

Операторы управления потоками, например, IF и LOOP, а также любые операции курсора, в режиме ретрансляции не реплицируются. Любые операторы в пределах цикла или управляющей структуры *реплицируются*.

Операции с курсорами не реплицируются. Операции вставки строк через курсор, обновление строк в курсоре и удаление строк через курсор в режиме ретрансляции не реплицируются.

Операторы SQL SET OPTION, внедренные статически, не реплицируются. Следующий оператор не реплицируется в режиме ретрансляции:

```
EXEC SQL SET OPTION . . .
```

Однако следующий динамический оператор SQL реплицируется:

```
EXEC SQL EXECUTE IMMEDIATE "SET OPTION . . . "
```

Пакетные операторы

Пакетные операторы (группа операторов, начинающихся с BEGIN и заканчивающихся END) в режиме ретрансляции не реплицируются. При попытке использовать пакетные операторы в режиме ретрансляции пользователь получает сообщение об ошибке.

Администрирование SQL Remote для Adaptive Server Enterprise

Об этой главе

В данной главе подробно рассматриваются вопросы настройки и управления системой для администраторов SQL Remote, использующих Adaptive Server Enterprise в качестве консолидированной базы данных.

Содержание

Раздел	Страница
Принципы работы Message Agent для Adaptive Server Enterprise	230
Работа с Message Agent	234
Сообщения об ошибках и их обработка	236
Управление журналом транзакций и резервным копированием Adaptive Server Enterprise	237
Внесение изменений в схему	240
Использование режима ретрансляции	241

Принципы работы Message Agent для Adaptive Server Enterprise

В данном разделе описывается процесс запуска и дальнейшей работы с Message Agent для Adaptive Server Enterprise. Существуют значительные различия в том, каким образом Message Agent работает с Adaptive Server Enterprise и Adaptive Server Anywhere. Это обусловлено тем, что данные серверы выполняют две разные роли.

☞ Для получения информации об особенностях Message Agent, общих для Adaptive Server Anywhere и Adaptive Server Enterprise, см. раздел "Работа с Message Agent" на стр. 193.

Message Agent – это
ssremote

Исполняемый файл Message Agent для Adaptive Server Enterprise следующий:

- ◆ Message Agent в операционных системах Windows - *ssremote.exe*
- ◆ Message Agent в операционных системах UNIX - *ssremote*.

Сканирование журнала транзакций

Message Agent сканирует журнал транзакций Adaptive Server Enterprise для сбора транзакций, предназначенных для отправки в удаленные базы данных. Он сохраняет эти транзакции в **очереди с сохранением**.

☞ Для получения дополнительной информации об очереди с сохранением см. раздел "Очередь с сохранением" на стр. 231. Для получения дополнительной информации об использовании в Message Agent очереди с сохранением см. раздел "Этапы выполнения операций Message Agent" на стр. 274.

SQL Remote Message Agent использует тот же интерфейс сканирования журнала транзакций, что и Adaptive Server Enterprise Log Transfer Manager (LTM). Adaptive Server Enterprise поддерживает **точку усечения**, являющуюся идентификатором самой старой страницы в журнале транзакций, необходимым для системы репликации.

SQL Remote Message Agent задает точку усечения сразу после сканирования транзакций в журнале и их выполнения в очереди с сохранением. Это позволяет команде **дампа транзакций** сразу восстановить наличие свободного места в журнале транзакций. Перед установкой точки усечения Message Agent не ожидает получения подтверждения от удаленных баз данных.

Необходимость запуска Message Agent для восстановления свободного места в журнале транзакций

Message Agent необходимо запускать достаточно часто, во избежание возникновения ситуаций отсутствия пространства в журнале транзакций. Команда дампа транзакций не восстанавливает пространство страниц после точки усечения.

Replication Server и
SQL Remote

Использование SQL Remote в базе данных Adaptive Server Enterprise, задействованной в системе Replication Server, может вызвать необходимость принятия во внимания некоторых других положений. Если параллельно с базой данных работает агент репликации (LTM), тогда SQL Remote Open Server необходимо использовать как дополнительный компонент. Работающие агенты репликации имеются на базах данных Adaptive Server Enterprise при следующих обстоятельствах:

- ◆ База данных участвует в системе Replication Server в качестве первичной базы данных, либо

- ◆ база данных участвует в системе Replication Server и использует вызовы асинхронной процедуры.

Если база данных задействована в системе Replication Server в качестве узла репликации и вызовы асинхронной процедуры не используются, тогда необходимости в SQL Remote Open Server нет.

☞ Для получения дополнительной информации об SQL Remote Open Server см. раздел "Использование SQL Remote с Replication Server" на стр. 243.

Очередь с сохранением

Message Agent для Adaptive Server Enterprise использует **очередь с сохранением** для хранения транзакций до их удаления. Очередь с сохранением – это пара таблиц базы данных, содержащих сообщения, которые могут понадобиться в системе репликации.

В SQL Remote для Adaptive Server Anywhere очередь с сохранением не используется.

Очередь с сохранением отличается от очереди с сохранением Replication Server

Sybase Replication Server также использует очереди с сохранением в качестве местоположения хранения сообщений репликации. Очереди с сохранением Replication Server и SQL Remote выполняют сходные функции, но они *не* являются одинаковыми.

Местоположение очереди с сохранением

Очередь с сохранением можно хранить в той же базе данных, что и реплицируемые таблицы, либо в другой базе данных. Хранение очереди с сохранением в отдельной базе данных усложняет резервное копирование и восстановление, однако может повысить производительность путем распределения рабочей нагрузки очереди с сохранением на отдельные устройства и/или на отдельный сервер Adaptive Server Enterprise.

Запрет на прямое внесение изменений в очередь с сохранением

Message Agent поддерживает очередь с сохранением в целях обеспечения нормального функционирования. Изменять очередь с сохранением напрямую не рекомендуется.

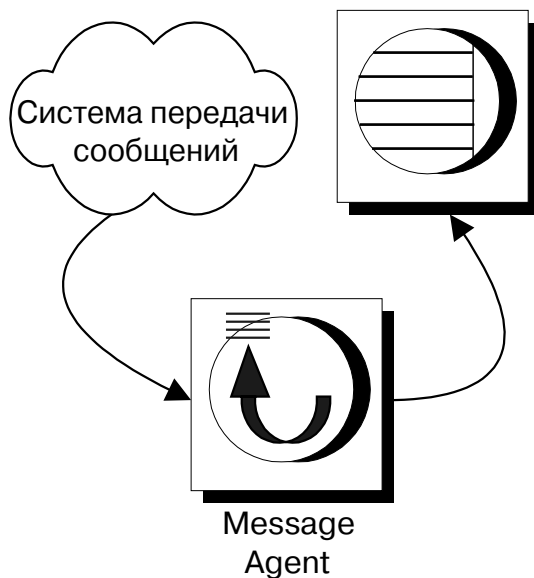
Очередь с сохранением состоит из набора таблиц, содержащих информацию обо всех транзакциях, сосканированных из журнала транзакций.

☞ Описание каждого из столбцов этих таблиц см. в разделе "Таблицы очереди с сохранением" на стр. 300.

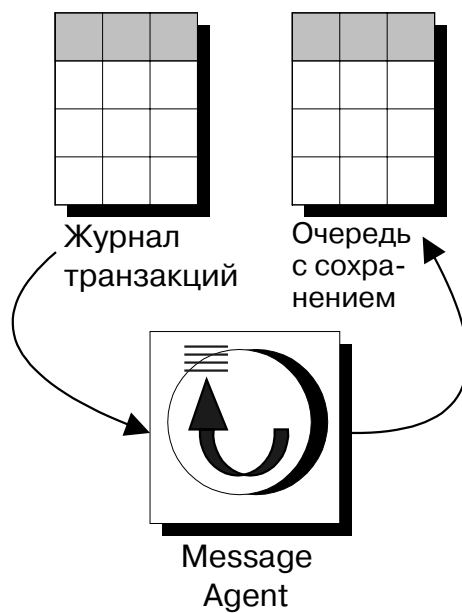
Этапы выполнения операций Message Agent

Операции Message Agent выполняются по следующим этапам:

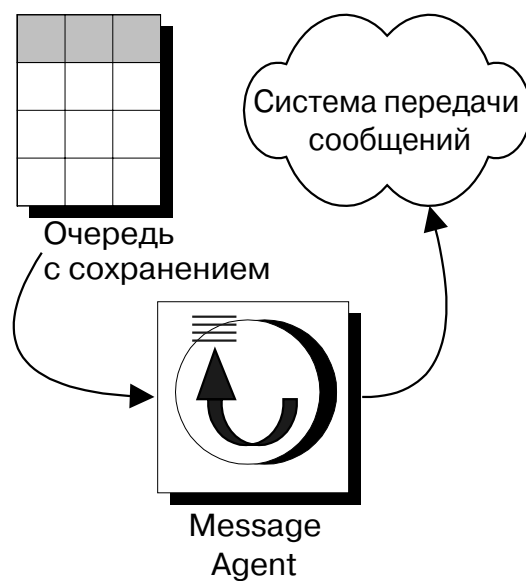
- ◆ **Получение сообщений.** На этом этапе Message Agent получает входящие сообщения и обрабатывает их для сервера Adaptive Server Enterprise.



- ◆ **Начальная загрузка очереди с сохранением.** На этом этапе Message Agent сканирует журнал транзакций Adaptive Server Enterprise в очередь с сохранением.



- ◆ **Отправка сообщений.** На этом этапе Message Agent формирует исходящие сообщения из очереди с сохранением.



Транзакции остаются в очереди с сохранением до получения подтверждений из всех удаленных баз данных. По получении подтверждения Message Agent автоматически удаляет транзакции из очереди с сохранением.

Message Agent не сканирует журнал транзакций базы данных, в которой находится очередь с сохранением, если она отличается от базы данных с системными таблицами SQL Remote.

☞ Для получения информации о запуске множественных Message Agent для выполнения этих задач см. раздел "Запуск множественных программ Message Agent" на стр. 234.

Работа с Message Agent

☞ В данном разделе описывается процесс запуска и дальнейшей работы с Message Agent для Adaptive Server Enterprise. Для получения информации об особенностях Message Agent, общих для Adaptive Server Anywhere и Adaptive Server Enterprise, см. раздел "Работа с Message Agent" на стр. 193.

Message Agent и безопасность репликации

В представленных ранее учебных разделах данного документа Message Agent запускался при помощи идентификатора пользователя с полномочиями системного администратора. Операции в сообщениях выполняются по идентификатору пользователя, указанному в строке подключения Message Agent; использование идентификатора системного администратора гарантирует наличие у данного пользователя всех необходимых полномочий для внесения изменений в конфигурацию системы.

На практике такой идентификатор пользователя применяться не будет, однако Message Agent должен запускаться при помощи идентификатора пользователя с ролью репликации. Роль репликации предоставляется при помощи следующего оператора:

```
sp_role 'grant', replication_role, имя_пользователя
```

Пользователь для Message Agent должен иметь полномочия на вставку, обновление и удаление во всех таблицах, участвующих в репликации, для обеспечения возможности применения внесенных изменений. Кроме этого, для используемого идентификатора пользователя Message Agent необходимо создать процедуру обработки ошибок репликации.

При настройке базы данных Adaptive Server Enterprise необходимо запустить сценарии *ssremote.sql* и *squeue.sql* с тем же именем пользователя, что применяется для Message Agent.

☞ Инструкции по настройке см. в разделе "Установка и настройка SQL Remote" на стр. 19.

Для того чтобы скрыть пароль для идентификатора пользователя Message Agent, можно сохранить параметры командной строки *ssremote* в файле и использовать *ssremote* с параметром *@filename*. Во избежание несанкционированного доступа к данному файлу можно использовать систему защиты файлов.

Запуск множественных программ Message Agent

Три этапа функционирования Message Agent описаны в разделе "Этапы выполнения операций Message Agent" на стр. 231. Здесь представлено краткое перечисление этих этапов:

- ◆ Получение сообщений;
- ◆ Сканирование журнала транзакций;
- ◆ Отправка сообщений.

Для выполнения операций на этих этапах может возникнуть необходимость в запуске отдельных копий Message Agent. Для определенного Message Agent можно задать этапы для выполнения в командной строке Message Agent.

Задание этапов для выполнения

Используются следующие параметры командной строки:

- ◆ **Получение.** Параметр `-r` командной строки направляет в запущенный Message Agent команду на получение сообщений. Для закрытия Message Agent после получения имеющихся сообщений наряду с параметром `-r` используйте параметр `-b`.
- ◆ **Сканирование журнала.** Параметр `-i` командной строки направляет в запущенный Message Agent команду сканирования журнала транзакций в очередь с сохранением.
- ◆ **Отправка.** Параметр `-s` командной строки направляет в запущенный Message Agent команду на отправку сообщений.
- ◆ **Все этапы.** Если не заданы параметры `-r`, `-i` или `-s`, тогда Message Agent выполняет все три этапа. В противном случае выполняются только указанные этапы.

Существует несколько обстоятельств, при которых может потребоваться запуск множественных Message Agent.

- ◆ **Подтверждение наличия свободного места для журнала транзакций.** Очень важно не заполнять журнал транзакций полностью. По этой причине необходимо достаточно часто сканировать журнал транзакций для контроля того, чтобы все записи, необходимые для SQL Remote, размещались в очереди с сохранением. Поэтому можно запустить Message Agent, который будет непрерывно сканировать журнал транзакций, даже если сообщения принимаются и отправляются в пакетном режиме.
- ◆ **Использование различных операционных систем.** При необходимости использования системы передачи сообщений, поддерживаемой одной операционной системой, для отправки и получения сообщений на этой платформе необходимо использовать Message Agent. При запущенном сканировании журнала на компьютере с UNIX это можно сделать запуском двух копий Message Agent.

Синхронизация
множественных
Message Agent

Операции двух или больше Message Agent синхронизируются через таблицу с именем **sr_marker**. В этой таблице имеется один столбец с именем **marker** и типом данных **datetime**.

Когда Message Agent необходимо дождаться транзакций для сканирования в очередь с сохранением, он обновляет **sr_marker** и дожидается его срабатывания в системе. Столбец в **sr_queue_state** также называется маркером (**marker**) и содержит самый последний маркер, который должен быть сосканирован из журнала транзакций.

Сообщения об ошибках и их обработка

В данном разделе описан процесс сообщения и обработки ошибок Message Agent.

Обработка ошибок по умолчанию

По умолчанию при возникновении ошибки Message Agent добавляет соответствующую запись в выводимой информации журнала. Message Agent отправляет выводимую информацию с этой записью в окно для просмотра пользователем или файл журнала. По умолчанию эта информация отображается только в окне; при использовании параметра `-o` также осуществляется вывод в файл журнала.

В выводимом журнале может содержаться больше информации, чем в окне. В журнал Message Agent входит следующее:

- ◆ Список обработанных сообщений;
- ◆ Список невыполненных операторов SQL;
- ◆ Список прочих ошибок.

Конфликты UPDATE не являются ошибками

Конфликты UPDATE не являются ошибками, поэтому отчеты о них не направляются в Message Agent.

Реализация процедур обработки ошибок

Помимо занесения в журнал сообщений об ошибках, SQL Remote позволяет выполнять некоторые другие процессы. Параметр `REPLICATION_ERROR` базы данных позволяет задать хранимую процедуру, вызываемую Message Agent при возникновении ошибок. По умолчанию никакие процедуры не вызываются.

В процедуре должен быть один аргумент типа `CHAR` или `VARCHAR`. Эта процедура вызывается дважды: один раз с сообщением об ошибке, и один раз – с оператором SQL, вызывающим ошибку.

Несмотря на то, что данный параметр позволяет отслеживать ошибки и контролировать их появление при репликации, возможность возникновения ошибок необходимо минимизировать на этапе настройки системы, поскольку данный параметр не обеспечивает устранения таких ошибок.

Например, данная процедура могла вставить ошибки в таблицу с текущим временем и идентификатором удаленного пользователя, после чего эту информацию можно реплицировать в консолидированную базу данных. Приложение в консолидированной базе данных при обнаружении ошибок генерирует сообщение об ошибке или отправляет администратору электронное письмо.

☞ Для получения информации о параметре `REPLICATION_ERROR` см. раздел "Параметры SQL Remote" на стр. 272.

Управление журналом транзакций и резервным копированием Adaptive Server Enterprise

Необходимо предотвратить возможность потери транзакций, которые были реплицированы на удаленные базы данных. Если транзакции, реплицированные в удаленные базы данных, утеряны, тогда последние не будут согласованы с консолидированной базой данных. В этой ситуации, возможно, придется повторно извлечь все удаленные базы данных.

Безопасность при отказе носителя журнала транзакций

Отказ носителей журналов транзакций может вызвать потерю выполненных транзакций. Если после сканирования журнала транзакций последние были отправлены в базы данных подписчиков, тогда эти базы данных будут содержать транзакции, утерянные из базы данных издателя. В этом случае базы данных будут несогласованными.

Необходимость ведения журнала транзакций

Для обеспечения безопасности при отказе носителя файла базы данных журнал транзакций необходим даже после того, как записи отсканированы в очередь с сохранением. Если база данных утеряна, то ее необходимо восстановить до момента отправки транзакций в удаленные базы данных.

Такое восстановление выполняется путем восстановления дампа базы данных и загрузкой дампов транзакций с целью обновления базы. Восстановленный дамп последней транзакции – это дамп активного журнала транзакций на момент отказа.

Защита от потери журнала транзакций

Есть два способа защиты от появления несогласованности в результате отказа носителя журнала транзакций:

- ◆ **Зеркальная копия журнала транзакций.** При зеркальном отображении устройства вся записываемая на него информация копируется на отдельное устройство.
- ◆ **Репликация только транзакций с резервной копией.** Существует параметр командной строки Message Agent, препятствующий отправке транзакций до создания их резервных копий.

Зеркальное отражение журнала транзакций

Единственный способ защиты от отказа носителей журнала транзакций – это создание зеркальной копии последнего.

Зеркальное отражение диска может обеспечить безостановочное восстановление в случае отказа носителей. Команда **disk mirror** дублирует устройство, на котором хранится база данных Adaptive Server Enterprise, т.е. все записи на устройстве копируются на отдельный физический носитель. При отказе одного из устройств все последние копии транзакций будут содержаться на другом носителе.

☞ Для получения информации о зеркальном отражении диска в Adaptive Server Enterprise см. главу "Зеркальное отражение носителей базы данных" (Mirroring Database Devices) в документе "Руководство по системному администрированию" (System Administration Guide) для Adaptive Server Enterprise.

Репликация только транзакций с резервной копией

Message Agent также предоставляет параметр командной строки (-u), который разрешает отправку только тех транзакций, для которых сделаны резервные копии. В Adaptive Server Enterprise это означает, что транзакции завершаются до выполнения последней команды **dump database** или команды **dump transaction**.

Выбор метода

Цель данной стратегии состоит в том, чтобы уменьшить необходимость повторного извлечения удаленных баз данных до приемлемого уровня. В крупных системах эта возможность должна как можно меньше, поскольку затраты на повторное извлечение (в плане времени простоя) очень высоки.

- ◆ Параметр `-u` командной строки Message Agent можно использовать вместо зеркального копирования журналов транзакций, когда не нужно проводить восстановление всех транзакций в консолидированной базе данных, поскольку сам процесс зеркального отображения также влечет за собой высокий расход ресурсов. Такой подход может быть оправдан при использовании в небольших системах или в системах, где в консолидированной базе данных нет локальных пользователей.
- ◆ Параметр `-u` командной строки Message Agent можно применять наряду с зеркальным отражением для обеспечения дополнительной защиты от полного отказа узла или двойного отказа носителей.

Особенности восстановления очереди с сохранением

Хранение очереди с сохранением в отдельной базе данных усложняет резервное копирование и восстановление, поскольку необходимо восстанавливать согласованные версии двух баз данных.

При нормальном восстановлении эти две базы данных автоматически становятся согласованными, хотя восстановление при отказе носителя требует определенной осторожности в действиях. При восстановлении дампов базы данных и дампов транзакций важно восстановить очередь с сохранением до точки согласования.

Для облегчения процесса восстановления при отказе носителей можно использовать следующие две процедуры в базах данных очереди с сохранением:

- ◆ **sp_queue_dump_database.** Эта процедура вызывается всякий раз, когда база данных дампа сканируется из журнала транзакций.
- ◆ **sp_queue_dump_transaction.** Эта процедура вызывается всякий раз, когда транзакция дампа сканируется из журнала транзакций.

Эти хранимые процедуры можно модифицировать для подачи в базу данных очереди с сохранением команд **dump database** и **dump transaction**.

Управление журналом транзакций

Интерфейс **log transfer** Adaptive Server Enterprise позволяет Message Agent сканировать журнал транзакций Adaptive Server Enterprise. При использовании этого интерфейса в журнале транзакций устанавливается **точка усечения**. Точка усечения препятствует Adaptive Server Enterprise повторно использовать страницы в журнале транзакций до их сканирования SSREMOTE. По этой причине DUMP TRANSACTION не обязательно выдает страницы журнала транзакций, находящиеся перед транзакцией, открытой одной из первых. DUMP TRANSACTION не выдает страницы журнала транзакций после точки усечения.

Инициализация точки усечения

Сценарий установки SQL Remote (*ssremote.sql*) инициализирует точку усечения при помощи следующей команды:

```
dbcc settrunc( 'ltm', 'valid' ).
```

Точку усечения можно удалить при помощи следующей команды:

```
dbcc settrunc( 'ltm', 'ignore' ).
```

Эта команда настраивает Adaptive Server Enterprise на игнорирование точки усечения, обеспечивая вывод страниц журнала транзакций за пределами точки усечения для повторного использования. Эту команду рекомендуется использовать только в случае отсутствия необходимости в репликации SQL Remote с базой данных и наличия необходимости в освобождении пространства на устройстве транзакции с командами DUMP TRANSACTION. Результатом продолжения работы с SQL Remote после игнорирования точки усечения будет

отказ репликации любых транзакций, находящихся на неотсканированных Message Agent страницах журнала транзакций, которые были высвобождены с помощью DUMP TRANSACTION.

Внесение изменений в схему

Изменения схемы таблиц, реплицируемых SQL Remote, должны вноситься в систему **в состоянии минимальной нагрузки**. Это означает следующее:

- ◆ **Никакие транзакции не реплицируются.** Транзакции, модифицирующие таблицы, которые должны быть изменены, не существуют. Все транзакции, модифицирующие изменяемые таблицы, необходимо отсканировать из журнала транзакций в очередь с сохранением перед тем, как будет изменена схема. Это выполняется обычным запуском Message Agent, либо при помощи параметров `-i`, `-b`. После завершения работы Message Agent в схему можно вносить изменения.
- ◆ **Message Agent.** Во время внесения изменений в схему Message Agent необходимо закрыть.
- ◆ **SQL Remote Open Server.** При работе с SQL Remote Open Server его необходимо закрыть во время внесения изменений в схему.

Изменения схемы включают в себя изменения в публикациях (например, добавление или изменения статей). Впрочем, создание или удаление подписок, а также добавление или удаление удаленных пользователей, не обязательно выполнять в состоянии минимальной нагрузки.

В журнале транзакций Adaptive Server Enterprise не содержится информации, регистрирующей изменения структуры таблиц: процесс сканирования журнала SQL Remote получает табличную структуру из системных таблиц Adaptive Server Enterprise.

Следовательно, Message Agent не может отсканировать операцию из журнала транзакций, которая имело место в старой структуре таблицы.

Информация, хранящаяся в очереди с сохранением до внесения изменений в схему, использует старые определения таблиц, а информация, сохраненная после изменения схемы, использует новые определения таблиц.

Режим ретрансляции можно использовать одновременно с внесением изменений в схему для обеспечения того, чтобы изменения схемы в удаленной базе данных происходили в корректном порядке.

Использование режима ретрансляции

Издатель консолидированной базы данных может напрямую войти в удаленные узлы при использовании режима ретрансляции, обеспечивающего стандартным операторам SQL доступ к удаленному узлу.

Определение получателей операторов ретрансляции

Адресаты ретрансляции определяются процедурами **sp_passthrough_user** и **sp_passthrough_subscription**. Выполнение любой из этих процедур определяет набор получателей для любых последующих операторов ретрансляции.

Выполнение любой процедуры **sp_passthrough_user** и **sp_passthrough_subscription** добавляет получателей в текущий список. Процедура **sp_passthrough_stop** сбрасывает ретрансляцию (т.е. очищает список получателей).

В Adaptive Server Enterprise процедура **sp_passthrough** никогда не выполняет операторы в консолидированной базе данных. Операторы ретрансляции SQL применяются только по отношению к удаленным базам данных.

Операторы ретрансляции

Для выполнения операторами ретрансляции SQL репликации вызывается процедура **sp_passthrough**.

Из-за ограничения VARCHAR (255) в Adaptive Server Enterprise длинные операторы SQL компонуются по частям. При выполнении вызовов **sp_passthrough_piece** осуществляется построение одного оператора SQL. При выполнении вызова **sp_passthrough** с последней частью построенный оператор будет реплицирован.

Примечания по использованию режима ретрансляции

- ◆ Операции ретрансляции необходимо всегда тестировать на тестовой базе данных с подписанной удаленной базой данных. Нельзя запускать непроверенные сценарии ретрансляции в действующей базе данных.

- ◆ Для объектов всегда должны указываться имена владельцев.

Операторы PASSTHROUGH не выполняются в удаленных базах данных по идентификатору пользователя. Поэтому имена объектов без указания имени владельца могут быть неправильно интерпретированы.

Модификации схемы

Интерфейс передачи журнала Adaptive Server Enterprise не содержит информации о количестве столбцов и типов данных столбцов в таблице. SSREMOTE получает эту информацию непосредственно из системных таблиц Adaptive Server Enterprise. По этой причине изменение таблицы с последующим сканированием операций, имевших место до ALTER TABLE, приведет к появлению ошибок. SSREMOTE должен задать "точку усечения" до всех операций в реплицируемых таблицах еще до того, как будет осуществлено изменение схемы. Необходимо исключить выполнение операций в реплицируемых таблицах между запуском SSREMOTE и изменениями схемы.

Использование SQL Remote с Replication Server

Об этой главе В данной главе описаны дополнительные компоненты, необходимые для использования SQL Remote в базе данных Adaptive Server Enterprise, задействованной в системе Replication Server.

Содержание

Раздел	Страница
Когда необходимо использовать SQL Remote Open Server	244
Архитектура систем Replication Server/SQL Remote	245
Установка и настройка SQL Remote Open Server	248
Конфигурирование Replication Server	250
Разное	252

Когда необходимо использовать SQL Remote Open Server

Message Agent для Adaptive Server Enterprise сканирует журнал транзакций Adaptive Server Enterprise для заполнения очереди с сохранением, как описано в разделе "Очередь с сохранением" на стр. 231. Сообщения SQL Remote формируются из транзакций в очереди с сохранением.

Для сканирования журнала транзакций Message Agent использует тот же интерфейс, что и Replication Agent для Adaptive Server Enterprise. Это означает, что Message Agent не может сканировать журнал транзакций базы данных Enterprise, являющейся первичным узлом в системе Replication Server (или узлом репликации, обеспечивающим асинхронные обновления первичных данных).

Если в базе данных Adaptive Server Enterprise имеется запущенный Replication Agent, тогда SQL Remote Open Server необходимо использовать в качестве дополнительного компонента. В этом случае SQL Remote настраивается так, чтобы Replication Server заполнял очередь с сохранением. SQL Remote Message Agent не сканирует журнал транзакций. Вместо этого SQL Remote Open Server получает транзакции от Replication Server. SQL Remote Open Server подвергает транзакции синтаксическому разбору и сохраняет в очереди с сохранением SQL Remote.

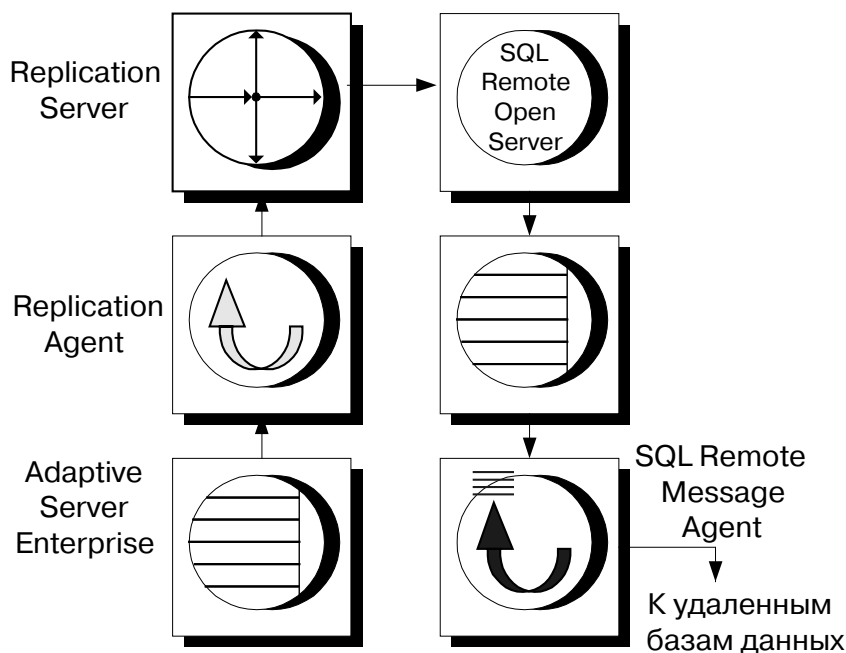
☞ В данной главе подразумевается наличие необходимых знаний по Replication Server. Дополнительную информацию см. в документации по Replication Server.

Требуемые компоненты рабочей среды Open Server

Компоненты рабочей среды Open Server не включены в SQL Remote. Для использования SQL Remote Open Server их необходимо отдельно приобрести в Sybase.

Архитектура систем Replication Server/SQL Remote

Архитектура в случае использования базы данных в качестве первичного узла Replication Server, а также в качестве базы данных SQL Remote, представлена в следующей схеме. Здесь проиллюстрирован случай, когда очередь с сохранением хранится в иной базе данных, нежели реплицируемые данные. Очередь с сохранением можно хранить в той же базе, что реплицируемые данные. Все подключения имеют тип клиент/сервер, поэтому эти компоненты можно запустить как на одном, так и на нескольких компьютерах.

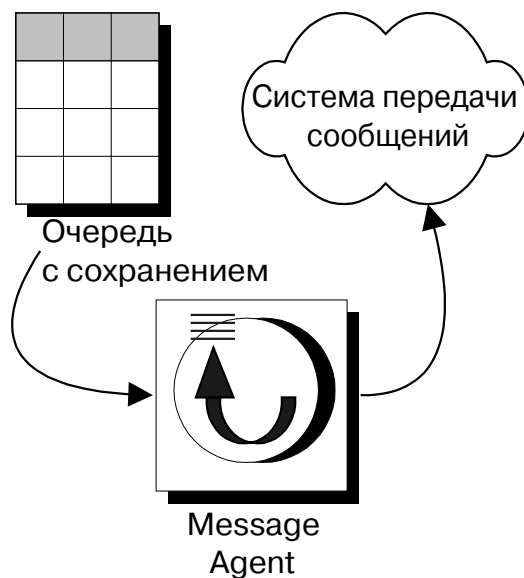


Соответствие компонентов

Содержимое
очереди с
сохранением

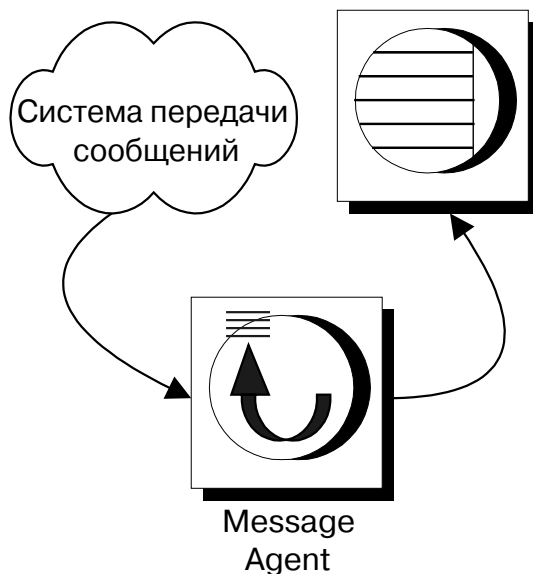
В системе Replication Server SQL Remote Open Server выступает в качестве базы данных репликации, поэтому в базе данных Adaptive Server Enterprise во всех таблицах, участвующих в репликации SQL Remote, а также в нескольких системных таблицах SQL Remote требуются определения репликации и подписки.

Все операции реплицируются на SQL Remote Open Server, на котором они хранятся в очереди с сохранением. В очереди с сохранением нет копий реплицируемых таблиц. Для формирования транзакций очередь с сохранением анализирует синтаксис вставок, обновлений и удалений. Все транзакции хранятся в столбце образа одной таблицы. Message Agent использует эти транзакции для формирования сообщений SQL Remote.



Входящие сообщения

Message Agent всегда обрабатывает входящие сообщения SQL Remote напрямую в Adaptive Server Enterprise. Message Agent не отправляет операции в Replication Server. Входящие сообщения обрабатываются напрямую в консолидированной базе данных, независимо от заполнения очереди с сохранением. Разрешение конфликтов выполняется аналогично.



Replication Server и SQL Remote

SQL Remote обеспечивает двунаправленную репликацию между консолидированной базой данных и удаленными базами данных. Replication Server выполняет однонаправленную репликацию из консолидированной базы данных в SQL Remote Open Server. Со стороны Replication Server транзакции, появляющиеся в удаленных базах данных SQL Remote, появляются как транзакции, инициированные в консолидированной базе данных SQL Remote.

Системные таблицы SQL Remote

SQL Remote Open Server необходима информация из системных таблиц SQL Remote, касающаяся публикаций и подписок. Open Server использует подключение к базе данных Adaptive Server Enterprise, содержащей эту информацию, для ее получения при запуске.

Если системные таблицы SQL Remote обновляются во время работы Open Server, тогда SQL Remote Open Server необходимо своевременно получить эту информацию. По этой причине некоторые системные таблицы SQL Remote необходимо отметить для репликации. Этот процесс описан в разделе "Установка и настройка SQL Remote Open Server" на стр. 292.

Исполняемый файл
SQL Remote Open
Server

Исполняемые файлы SQL Remote Open Server следующие:

- ◆ В операционных системах Windows исполняемый файл SQL Remote Open Server - *ssqueue.exe*.
- ◆ В операционных системах UNIX исполняемый файл SQL Remote Open Server - *ssqueue*.

Установка и настройка SQL Remote Open Server

В данном разделе описывается процесс настройки системы SQL Remote с SQL Remote Open Server. Данная процедура зависит от того, сохранена очередь с сохранением SQL Remote в отдельной базе данных Adaptive Server Enterprise, нежели реплицируемые таблицы, или в той же базе данных Adaptive Server Enterprise.

☞ Для получения дополнительной информации о местоположении очереди с сохранением см. раздел "Очередь с сохранением" на стр. 231.

Начальные копии данных

Процедура настройки предполагает использование утилиты извлечения для создания начальной копии данных в каждой удаленной базе данных. Необходимо убедиться в том, что для этой цели не используется средство материализации Replication Server.

Процедура установки SQL Remote Open Server содержит два этапа:

- ◆ **Подготовка к установке SQL Remote.** Действия на этом этапе зависят от наличия или отсутствия инсталляции SQL Remote.
- ◆ **Добавление SQL Remote Open Server к системе.** Действия на этом этапе выполняются одинаково независимо от предыдущих инсталляций.

❖ Процедура подготовки к установке SQL Remote при наличии инсталляции SQL Remote

- 1 В первичной базе данных с минимальной нагрузкой воспользуйтесь Message Agent для сканирования оставшихся транзакций в очередь с сохранением.
База данных с минимальной нагрузкой – база данных, где не запущен ни Message Agent, ни SQL Remote Open Server, и где не выполняется репликация транзакций.
- 2 Для обновления ПО SQL Remote в узле консолидированной базы данных выполняйте шаги, описанные в разделе "Обновление SQL Remote для Adaptive Server Enterprise" на стр. 22.
- 3 Отмените точку усечения Message Agent в консолидированной базе данных, используя следующую команду:

```
dbcc settrunc('ltm', 'ignore')
```
- 4 В базе данных очереди с сохранением выполните хранимую процедуру `sp_queue_log_transfer_reset`.

❖ Процедура подготовки к установке SQL Remote при отсутствии инсталляции

- 1 Сконфигурируйте систему SQL Remote, как описано в разделе "Установка и настройка SQL Remote" на стр. 19.
- 2 Создайте в этой точке публикации SQL Remote и подписки. Информацию об этой процедуре см. в разделе "Настройка SQL Remote для Adaptive Server Enterprise" на стр. 121.
- 3 Извлеките удаленные базы данных. Информацию об этой процедуре см. в разделе "Использование утилиты извлечения" на стр. 165.

Теперь все готово для установки SQL Remote Open Server.

❖ Процедура установки SQL Remote Open Server

- 1 Если очередь с сохранением SQL Remote находится в отдельной базе данных:

- ◆ Установите базу данных очереди с сохранением в качестве базы данных репликации в настройках Replication Server. При этом будут созданы таблицы и процедуры, необходимые Replication Server, например, **rs_lastcommit**.
 - ◆ Удалите подключение Replication Server к базе данных очереди с сохранением.
- 2 Добавьте запись в файлы интерфейсов для SQL Remote Open Server. Имя по умолчанию, используемое в командной строке SQL Remote Open Server, - **SSQueue**.
 - 3 Запустите SQL Remote Open Server.
 - 4 Создайте подключение Replication Server к SQL Remote Open Server. Идентификатор пользователя и пароль для этого подключения должны соответствовать идентификатору пользователя и паролю, указанному в командной строке SQL Remote Open Server для подключения очереди с сохранением (т.е. параметр `-sq` или `-s`, если параметр `-sq` не задан).

Конфигурирование Replication Server

На данном этапе необходимо сконфигурировать Replication Server для данного подключения. Описание процедуры см. в разделе "Конфигурирование Replication Server" на стр. 250.

- 5 Определите, активизируйте и проверьте определения репликации и подписки Replication Server для таблиц SQL Remote **sr_marker**, **sr_remoteuser**, **sr_subscription** и **sr_passthrough**. Сценарий **ssremote.rs** является демонстрационным сценарием для выполнения этой задачи. Имена базы данных и сервера в сценарии необходимо отредактировать в соответствии с имеющимися именами.

Если системные таблицы SQL Remote содержат какие-либо данные, во избежание материализации создайте определения репликации.

☞ Для получения информации о создании определений репликации без материализации см. документ "*Руководство по администрированию Replication Server*" (*Replication Server Administration Guide*). В разделе "Массовая материализация" (Bulk Materialization) главы 10 ("Управление подписками" - *Managing Subscriptions*) описывается настройка Replication Server в тех случаях, когда в удаленной базе данных имеются данные.

- 6 Определите, активизируйте и проверьте определения репликации и подписки для таблиц в базе данных, предназначенной для репликации SQL Remote. Они должны быть созданы без материализации.

Конфигурирование Replication Server

В данном разделе описан процесс конфигурирования Replication Server для использования с SQL Remote Open Server.

Подключение Replication Server к SQL Remote Open Server должно иметь несколько наборов установленных параметров конфигурации.

Задание параметра `dsi_xact_group_size`

По умолчанию Replication Server группирует множественные транзакции в транзакции большего объема. Параметр `dsi_xact_group_size` регулирует максимальный размер сгруппированной транзакции.

Для отключения функции группирования транзакций значение параметра `dsi_xact_group_size` должен быть `-1`. Транзакции, появляющиеся из разных удаленных баз данных в системе SQL Remote, не должны группироваться.

Процедура задания параметра

Данный параметр можно задать с помощью следующего оператора:

```
CONFIGURE CONNECTION TO "ssqueue_server"  
SET dsi_xact_group_size TO '-1'
```

Задание параметра `dsi_num_threads`

SQL Remote Open Server не поддерживает множественные потоки DSI. Конфигурация Replication Server не должна предусматривать использования потоков DSI для подключений SQL Remote.

Создание определений репликации для данных SQL Remote

Определения репликации для таблиц, реплицируемых SQL Remote, должны иметь соответствующие характеристики. Эти характеристики описываются в данном разделе.

В некоторых обстоятельствах SQL Remote реплицирует операцию UPDATE как INSERT или DELETE (см. раздел "Репликация операторов обновления" на стр. 77). В документации по Replication Server это упоминается как **перемещение подписки** (subscription migration). Для репликации UPDATE как INSERT SQL Remote требует полного исходного образа строки. Это означает, что Replication Server должен определить значения каждого столбца в разделе WHERE любого UPDATE таблицы, которую, возможно, потребуется реплицировать как INSERT.

Для достижения этого самым простым способом является перечисление всех столбцов в первичном ключе определения репликации. Это вынуждает Replication Server включать каждый столбец в раздел WHERE каждого обновления. Для предотвращения включения каждого столбца в раздел SET обновления в определениях репликации можно использовать REPLICATE MINIMAL COLUMNS.

Текст и столбцы образов

Replication Server не принимает столбцы TEXT или IMAGE в первичном ключе определения репликации. В PRIMARY KEY определения репликации необходимо включить все столбцы, за исключением столбцов TEXT и IMAGE, но задать столбцы TEXT и IMAGE в разделе ALWAYS_REPLICATE. В определении репликации необходимо использовать REPLICATE ALL COLUMNS вместо REPLICATE MINIMAL COLUMNS. Это вынуждает Replication Server отправлять

исходный образ столбцов TEXT и IMAGE на SQL Remote Open Server всякий раз при обнаружении обновления.

Использование
стиля данных
dsi_sql_data_style

Replication Server 11.5 имеет новый **dsi_sql_data_style** для SQL Remote. Этот стиль данных автоматически включает все столбцы в раздел WHERE каждого UPDATE. Заносить все столбцы в PRIMARY KEY определения репликации необходимости нет. Использование определения репликации REPLICATE MINIMAL COLUMNS не дает Replication Server сохранять полный исходный образ обновляемых строк, поэтому SQL Remote **dsi_sql_data_style** не будет работать с REPLICATE MINIMAL COLUMNS.

Приостановка и перезапуск подключения

После конфигурирования подключения Replication Server к SQL Remote Open Server необходимо приостановить и возобновить подключение для вступления в силу настроек параметров. Эта задача выполняется при помощи следующей команды:

```
suspend connection to сервер_ssqueue
go

resume connection to сервер_ssqueue
go
```

Разное

В данном разделе представлены разные вопросы использования SQL Remote с Replication Server.

Запуск Message Agent. Message Agent должен запускаться с параметрами командной строки для отправки и приема (-r и -s). При этом Message Agent не будет сканировать журнал транзакций. Если Message Agent пытается сканировать журнал транзакций во время работы Replication Agent, то результатом станет ошибка при попытке зарезервировать "контекст передачи журнала" (log transfer context).

Вызовы процедур в SQL Remote Open Server. SQL Remote Open Server передает все принимаемые вызовы процедур с Replication Server в базу данных очереди с сохранением. Например, **rs_get_lastcommit** и **rs_update_lastcommit** выполняются в базе данных очереди с сохранением.

Скоординированные дампы. Replication Server обеспечивают механизм координирования дампов базы данных и дампов журнала транзакций между основной базой данных и базой данных очереди с сохранением. Функциональные строки **rs_dumpdb** и **rs_dumptran** можно использовать для реализации скоординированных дампов базы данных очереди с сохранением. Подробная информация представлена в документации к Replication Server.

Изменения схемы. При внесении изменений в схему в системе SQL Remote их необходимо выполнять при минимальной нагрузке системы. При этом SQL Remote Open Server необходимо закрыть.

Справочная информация

В данной части содержатся справочные материалы по SQL Remote.



Справочник по утилитам и параметрам

Об этой главе В данной главе содержится справочный материал по утилитам SQL Remote и параметрам баз данных SQL Remote.

В ней также описаны клиентские хранимые процедуры перехватчиков событий, которые могут использоваться для настройки процессов репликации.

Содержание

Раздел	Страница
Message Agent	256
Утилита извлечения базы данных	264
SQL Remote Open Server	270
Параметры SQL Remote	272
Процедуры перехватчиков событий SQL Remote	276

Message Agent

Назначение

Отправка и получение сообщений SQL Remote, а также поддержка системы отслеживания сообщений для обеспечения доставки сообщений.

Синтаксис

{ **dbremote** | **ssremote** } [*параметры*] [*папка*]

Параметры

Параметр	Описание
@ <i>имя-файла</i>	Чтение входных параметров из файла конфигурации
@ <i>envvar</i>	Чтение входных параметров из переменных среды
-a	Запрет обработки полученных транзакций
-b	Выполнение в пакетном режиме
-c " <i>ключевое слово=значение; ...</i> "	Указание параметров подключения к базам данных
-cq " <i>ключевое слово=значение; ...</i> "	Указание параметров подключения к базам данных для очереди с сохранением (только для Adaptive Server Enterprise)
-dl	Отображение сообщений в журнале на экране
-ek <i>ключ</i>	Определение ключа шифрования
-ep	Запрос ключа шифрования
-e <i>строка-языка</i>	Установка языка ("locale", только для Adaptive Server Enterprise)
-fq	Полное сканирование очереди с сохранением при отправке сообщений (только для Adaptive Server Enterprise)
-g <i>n</i>	Группирование транзакций, состоящих менее чем из <i>n</i> операций.
-i	Сканирование транзакций из журнала транзакций в очередь с сохранением (только для Adaptive Server Enterprise).
-k	Закрытие окна по завершении
-l <i>длина</i>	Максимальная длина сообщения
-m <i>размер</i>	Максимальный объем памяти, используемый для создания сообщений.
-o <i>файл</i>	Вывод сообщений в файл
-os <i>размер</i>	Максимальный размер файла регистрации выходных сообщений
-ot <i>файл</i>	Усечение файла и регистрация выходных сообщений
-p	Не очищать сообщения
-q	Выполнение в свернутом окне
-r	Получение сообщений
-rd <i>минуты</i>	Период опроса для входящих сообщений
-ro <i>имя-файла</i>	Регистрация удаленного выхода в файл
-rp <i>число</i>	Число опросов о получении прежде чем сообщение будет считаться потерянным

Параметр	Описание
-rt <i>имя-файла</i>	Усечение и запись выводимой информации удаленной БД в файл
-ru <i>время</i>	Период ожидания перед повторным сканированием журнала по получении повторной отправки
-s	Отправка сообщений
-sd <i>время</i>	Период посылки опроса
-t	Репликация всех триггеров (только для Adaptive Server Anywhere)
-u	Обработка только транзакций с резервными копиями
-ud	Для платформ UNIX: выполнить как демон
-v	Работа в расширенном режиме
-w <i>n</i>	Число рабочих потоков для обработки входящих сообщений (не в случаях NetWare или Windows CE)
-x [<i>размер</i>]	Переименование и перезапуск журнала транзакций (только Adaptive Server Anywhere)
directory	Папка, в которой содержатся старые журналы транзакций (только для Adaptive Server Anywhere)

Описание

Message Agent отправляет и обрабатывает сообщения для репликации SQL Remote, а также поддерживает работу системы отслеживания сообщений для контроля доставки сообщений.

Имеются исполняемые файлы Message Agent со следующими именами:

- ◆ **dbremote.** Message Agent для Adaptive Server Anywhere.
- ◆ **ssremote.** Message Agent для Adaptive Server Enterprise.

Можно также выполнить Message Agent из собственного приложения путем вызова библиотеки DBTools. Для получения дополнительной информации см. файл *dbrrmt.h* в подпапке *h* папки установки SQL Remote.

В случае Adaptive Server Anywhere пользователи, указываемые в командах Message Agent, должны иметь полномочия REMOTE DBA или DBA. В случае Adaptive Server Enterprise идентификаторы пользователей должны иметь роль репликации.

Дополнительный параметр *directory* определяет папку, в которой содержатся старые журналы транзакций, что обеспечивает доступ Message Agent к событиям до запуска текущего журнала.

Message Agent использует множество подключений к базе данных. Для вывода списка подключений см. раздел "Подключения, используемые Message Agent" на стр. 194.

☞ Для получения информации о полномочиях REMOTE DBA см. раздел "Message Agent и безопасность репликации" на стр. 211.

@имя-файла Чтение входных параметров из указанного файла.

В файлах могут содержаться концы строк и любые наборы параметров. Например, в приведенном ниже командном файле содержится набор параметров для Message Agent, который выполняется при размере кэша 4 Мб, осуществляет только передачу сообщений и выполняет подключение к базе данных **field** на сервере **myserver**:

```
-m 4096
-s
-c "eng=myserver; dbn=field; uid=sa; pwd=sysadmin"
```

Подробное описание параметров

Если этот конфигурационный файл сохранен как `c:\config.txt`, то он может следующим образом использоваться в команде:

```
ssremote @c:\config.txt
```

или

```
dbremote @c:\config.txt
```

@переменная-среды Чтение входных параметров из указанной переменной среды.

В переменной среды может содержаться любой набор параметров. Например, первый из двух приведенных ниже операторов задает переменную среды, которая содержит набор параметров для сервера базы данных, который работает при размере кэша 4 Мб, осуществляет только прием сообщений и выполняет подключение к базе данных **field** на сервере **myserver**. Весь оператор **set** должен быть введен одной строкой:

```
set envvar=-m 4096 -r
-c "eng=myserver;dbn=field;uid=sa;pwd=sysadmin"
ssremote @envvar
```

-a Обработка полученных сообщений (сообщений, находящихся в почтовый ящик для входящих сообщений) без их обработки в базе данных. Использование этого параметра вместе с параметрами **-v** (для расширенного вывода) и **-p** (без очистки сообщений) помогает при обнаружении проблем с входящими сообщениями. При использовании данного параметра без **-p** почтовый ящик для входящих сообщений очищается без обработки сообщений, что требуется при повторной активизации подписки.

-b Выполнение в пакетном режиме. В этом режиме Message Agent обрабатывает входящие сообщения, однократно сканирует журнал транзакций и обрабатывает исходящие сообщения, а затем завершает свою работу.

-c "параметр=значение; ... " Определение параметров подключения. Если этот параметр не определен, то в Adaptive Server Anywhere используется переменная среды `SQLCONNECT`.

Например, нижеприведенный оператор выполняет `dbremote` для файла базы данных `c:\Program Files\Sybase\SQL Anywhere 8\asdemo.db`, устанавливая подключение при использовании идентификатора пользователя **DBA** и пароля **SQL**:

```
dbremote -c "uid=DBA;pwd=SQL;dbf=c:\Program
Files\Sybase\SQL Anywhere 8\asdemo.db"
```

Message Agent должен выполняться пользователями, имеющими полномочия `REMOTE DBA` или полномочия `DBA`.

☞ Для получения информации относительно полномочий `REMOTE DBA` см. раздел "Message Agent и безопасность репликации" на стр. 211.

Message Agent для Adaptive Server Anywhere поддерживает полный набор параметров подключения Adaptive Server Anywhere. Message Agent для Adaptive Server Enterprise поддерживает следующие параметры подключения:

Параметр	Описание
UID	Имя пользователя
PWD	Пароль
DBN	Имя базы данных (необязательный параметр) При отсутствии этого параметра подключение выполняется к базе данных по умолчанию для указанного имени пользователя.
ENG	Имя Adaptive Server Enterprise.

-sq "параметр=значение; ... " Определение параметров подключения для очереди с сохранением. Данный параметр применим только в Adaptive Server Enterprise. Если этот параметр не указан, значения устанавливаются по умолчанию как значения, указанные в `-s`.

-dl Отображение сообщений в окне Message Agent или в командной строке, а также в файле журнала, если это указано.

Определение ключа шифрования (-ек) Этот параметр позволяет непосредственно в командной строке определять ключ шифрования для строго зашифрованных баз данных. В случае строго зашифрованных баз данных для использования баз данных или журналов транзакций, включая неактивные журналы транзакций, необходимо любым способом обеспечить ключ шифрования. В случае строго зашифрованных баз данных необходимо указать либо `-ек`, либо `-ер`, но не оба параметра одновременно. При отсутствии указания ключа для строго зашифрованных баз данных команда не будет выполнена.

Запрос на ввод ключа шифрования (-ер) Этот параметр позволяет указать, что требуется запрос на ввод ключа шифрования. Данный параметр вызывает открытие диалогового окна, в котором вводится ключ шифрования. Это обеспечивает дополнительную меру безопасности, так как ключ шифрования никогда не показывается в явном текстовом виде. В случае строго зашифрованных баз данных необходимо указать либо `-ек`, либо `-ер`, но не оба параметра одновременно. При отсутствии указания ключа для строго зашифрованных баз данных команда не будет выполнена.

-е строка-языка Этот параметр применим только в Adaptive Server Enterprise. Определяет информацию о языке Adaptive Server Enterprise. Строка языка имеет следующий формат:

```
"название_языка, кодовая_таблицы [, порядок_сортировки]"
```

По умолчанию Message Agent использует значение языка, определенное в файле `sybase\locales\locales.dat`.

Если *название_языка* и *кодовая_таблицы* не указаны, Message Agent получает их от Adaptive Server Enterprise. Если не указан *порядок_сортировки*, Message Agent использует двоичный порядок сортировки (по значениям байтов).

-fq Данный параметр используется только в Adaptive Server Enterprise. Использование этого параметра дает возможность полного сканирования очереди с сохранением при отправке сообщений, начиная с самого старого значения `confirm_sent` в таблице `sr_remoteuser`.

Это средство предназначено для неперiodического использования и позволяет производить очистку очереди с сохранением большого объема. Если, например, отдельные пользователи долго не подтверждают получение сообщений, очередь с сохранением может сильно увеличиться. Однако при выполнении `-fq` могут удалиться более актуальные подтверждения от других пользователей, даже если они являются более новыми, чем установленная по умолчанию точка отсечения сообщений.

-g n Инструкция для Message Agent о необходимости сгруппировать транзакции, содержащие вместе с последующими транзакциями менее чем *n* операций. Значение по умолчанию - двадцать операций. Увеличение значения *n* может ускорить обработку входящих сообщений за счет уменьшения количества подтверждений. Однако из-за увеличения размеров транзакций это может также вызвать появление блокировок и взаимоблокировок.

-i Сканирование транзакций из журнала транзакций в очередь с сохранением. Этот параметр доступен только для Adaptive Server Enterprise. Он используется при необходимости запуска отдельной копии Message Agent для сканирования журнала транзакций и для отправки и получения сообщений.

Если не указан ни один из параметров `-r`, `-i`, или `-s`, Message Agent выполняет все три операции. В противном случае выполняются только указанные операции.

☞ Для получения дополнительной информации см. раздел "Запуск множественных программ Message Agent" на стр. 277.

-к Закрытие окна по завершении, если данный параметр используется совместно с параметром -о.

-l длина Определяет максимальную длину в байтах каждого сообщения, которое будет отправлено. Более длинные транзакции разбиваются на более чем одно сообщение. Значение по умолчанию - 50000 байтов, а минимальная длина - 10000.

Предостережение

Максимальная длина сообщений должна быть на всех узлах системы одной и той же.

Для платформ с ограниченным выделением памяти значение должно быть меньше объема памяти, максимально выделяемого операционной системой.

-m размер Определяет максимальный объем памяти, используемый Message Agent для создания сообщений и кэширования входящих сообщений. Разрешенный объем может быть определен как *n* (в байтах), *nK* или *nM*. Значение по умолчанию - 2048 Кб (2 Мб).

Отдельные сообщения для каждой удаленной базы данных создаются одновременно в случае получения всеми удаленными базами данных реплицированных уникальных поднаборов операций. Для группы удаленных пользователей, получающих те же самые операции, создается только одно сообщение. Когда объем используемой памяти превышает значение **-m**, сообщения отправляются до достижения ими максимального размера (определенного параметром **-l**).

По поступлении сообщения сохраняются в памяти Message Agent до их обработки. Повторное считывание сообщений, поступающих от системы передачи сообщений не по порядку, может в случае крупных систем понизить производительность. Кэширование же сообщений предотвращает такое повторное считывание. При превышении выделенного объема памяти, заданного при помощи параметра **-m**, сообщения удаляются, начиная с самых старых сообщений.

-о Добавление выводимой информации в конец файла журнала. По умолчанию информация выводится на экран.

-os Определение максимального размера файла для регистрации выводимых сообщений. Разрешенный объем может быть определен как *n* (в байтах), *nK* (Кб) или *nM* (Мб). По умолчанию никаких ограничений нет; нижний предел составляет 10000 байтов.

Текущий размер файла проверяется перед выводом в файл сообщений журнала SQL Remote. Если сообщение их журнала приведет к превышению размера файла над указанным размером, то SQL Remote переименовывает файл вывода следующим образом: *ggmmddxx.dbr* (для dbremote) или *ggmmddxx.ssr* (для ssremote), где *xx* - последовательные символы в пределах от AA до ZZ, а *ggmmdd* соответствуют текущему году, месяцу и дню.

Этот параметр позволяет вручную удалять старые файлы журналов и освобождать дисковое пространство, если Message Agent может непрерывно работать в течение долгого времени.

-ot Усечение файла журнала с последующим добавлением выводимых сообщений в его конец. По умолчанию в таких случаях информация выводится на экран.

-р Обработка сообщений без их очистки.

-q Только для оконных операционных систем: выполнение Message Agent в свернутом окне.

-r Получение сообщений. Если не указан ни один из параметров `-r`, `-i` или `-s`, Message Agent выполняет все три операции. В противном случае выполняются только указанные операции.

Если Message Agent вызван с указанием параметра `-r`, он работает непрерывно. Для завершения работы Message Agent после получения сообщений, используйте параметр `-b` в дополнение к `-r`.

-rd время По умолчанию Message Agent запрашивает входящие сообщения каждую минуту. Этот параметр (`rd` означает "receive delay", **задержка получения**) позволяет конфигурировать периодичность опроса, что полезно при высоких затратах на выполнение опросов.

Для указания секунд после чисел можно использовать суффикс `s`, который используется при необходимости более частого выполнения опроса. Например:

```
dbremote -rd 30s
```

Опрос выполняется каждые тридцать секунд.

☞ Для получения дополнительной информации по опросам см. раздел "Настройка опроса входящих сообщений" на стр. 198.

-ro Этот параметр используется на консолидированных узлах. Если удаленные базы данных сконфигурированы для отправки выводимой информации журнала в консолидированную базу данных, указание данного параметра приводит к тому, что информация записывается в файл. Параметр предназначается для помощи администраторам при поиске и устранении ошибок в удаленных узлах.

☞ Для получения дополнительной информации см. раздел "Устранение неисправностей в удаленных узлах" на стр. 196.

-rp При непрерывной работе Message Agent производит опрос сообщений через некоторые интервалы времени. При отсутствии сообщения после указанного количества повторов опроса (по умолчанию одного) Message Agent предполагает, что сообщение утеряно, и запрашивает его повторную передачу. На медленных системах передачи сообщений это может привести к появлению большого количества ненужных запросов на повторную передачу. Данный параметр позволяет задать число опросов до отправки запроса на повторную передачу, что сократит количество таких запросов.

☞ Для получения дополнительной информации о конфигурировании этого параметра см. раздел "Настройка опроса входящих сообщений" на стр. 198.

-rt Этот параметр используется на консолидированных узлах. Он идентичен параметру `-ro` за исключением того, что файл усекается при запуске.

-ru Управление **срочностью повторной передачи**. Это время между обнаружением запроса на повторную передачу и началом выполнения этого запроса Message Agent. Этот параметр используется при сборе Message Agent запросов на повторную передачу от множественных пользователей до повторного сканирования журнала. Применимы любые из следующих единиц измерения времени: {s = секунды; m = минуты; h = часы; d = дни }

-s Отправка сообщений. Если ни один из параметров `-r`, `-i` или `-s` не указан, Message Agent выполняет все три операции. В противном случае выполняются только указанные операции.

-sd время Управление **задержкой при отправке**, которая является временем ожидания между запросами на отправку большого количества данных журнала транзакций.

-t Репликация всех действий триггеров. При использовании этого параметра необходима уверенность в том, что действия триггеров в удаленных базах данных не выполняются дважды: первый раз - триггером, запущенным на удаленном узле, и второй - явным выполнением реплицированных действий из консолидированной базы данных.

Во избежание повторного выполнения действий триггеров можно обернуть оператор IF CURRENT REMOTE USER IS NULL ... END IF вокруг тела триггеров. Этот параметр доступен только в Adaptive Server Anywhere.

-u Обработка только транзакций с имеющимися резервными копиями. Данный параметр запрещает Message Agent обработку транзакций, поступивших с момента последнего резервного копирования. Использование этого параметра запрещает отправку исходящих транзакций и подтверждений входящих транзакций до их резервного копирования.

В Adaptive Server Anywhere это означает, что будут обрабатываться только транзакции из переименованных журналов. Применительно к Adaptive Server Enterprise, это означает, что будут обрабатываться только транзакции, совершенные перед последним оператором **дампа базы данных** или **дампа транзакций**.

-ud Указание параметра `-ud` на платформах UNIX позволяет запустить Message Agent как демон.

При выполнении Message Agent как демона необходимо также указать параметр `-o` или `-ot` для записи в выводимой информации журнала.

При запуске Message Agent как демона и использовании систем передачи сообщений FTP или SMTP необходимо сохранить параметры системы передачи сообщений в базе данных, потому что при таком запуске Message Agent не запрашивает у пользователя эти параметры.

☞ Для получения информации относительно параметров системы передачи сообщений см. раздел "Установка параметров управления типами сообщений" на стр. 186.

-v Расширенный вывод. Этот параметр отображает на экране операторы SQL, содержащиеся в сообщениях, и, если указаны параметры `-o` или `-ot`, в файле журнала.

-w n Количество рабочих потоков, применяемых для обработки входящих сообщений. Величиной по умолчанию является ноль, что означает, что для всех сообщений используется основной (единственный) поток. Значение 1 (единица) означает, что имеется один поток, получающий сообщения от системы передачи сообщений, и еще один поток, обрабатывающий сообщения в базе данных.

Параметр `-w` позволяет увеличить пропускную способность при приеме сообщений по мере обновлений аппаратных средств. Помещение консолидированных баз данных на устройства, которые могут выполнять несколько параллельных операций (массив RAID с распределенным дисковым массивом), повышает пропускную способность при приеме сообщений. Множественные процессоры в компьютерах с запущенным Message Agent также могут увеличить пропускную способность при приеме сообщений.

При применении параметра `-w` для аппаратных средств, которые не могут выполнять несколько параллельных операций, их производительность увеличивается несущественно.

Входящие сообщения от одной удаленной базы данных никогда не обрабатываются несколькими потоками. Сообщения от одной удаленной базы данных всегда обрабатываются последовательно в правильном порядке.

-x Переименование и перезапуск журнала транзакций после его сканирования на предмет исходящих сообщений. В некоторых обстоятельствах репликация данных в консолидированную базу данных может производиться вместо резервного копирования удаленных баз данных или переименования журнала транзакций при отключенном сервере базы данных. Данный параметр доступен только в Adaptive Server Anywhere.

При указании дополнительного спецификатора *размер (size)* переименование журнала транзакций происходит только в том случае, если его размер превышает

**Параметры
управления
системой передачи
сообщений**

указанный. Разрешенный размер может быть указан как *n* (в байтах), *nK* или *nM*. Значением по умолчанию является 0.

SQL Remote использует несколько установок реестра для управления поведением системы передачи сообщений.

Параметры управления системами передачи сообщений хранятся в следующих местоположениях:

- ◆ **Windows.** В реестре в следующем разделе:

```
\\KEY_CURRENT_USER
  \Software
    \Sybase
      \SQL Remote
```

- ◆ **NetWare.** Необходимо создать файл с именем *dbremote.ini* в каталоге *sys:\system* для хранения установок каталога системы FILE.

☞ Для получения списка параметров реестра см. соответствующий подраздел для каждой системы передачи сообщений в разделе "Управление типами сообщений" на стр. 183.

Утилита извлечения базы данных

Утилиту извлечения удаленной базы данных можно вызвать следующими способами:

- ◆ Из Sybase Central для работы в интерактивном режиме;
- ◆ Из системной командной строки путем запуска утилит *ssxtract* или *dbxtract*. Этот метод используется при объединении в пакетный или командный файл.
ssxtract - утилита извлечения для Adaptive Server Enterprise, *dbxtract* - утилита извлечения для Adaptive Server Anywhere.

По умолчанию утилита извлечения выполняется на нулевом уровне изоляции. При извлечении базы данных с активного сервера утилиту необходимо запускать на третьем уровне изоляции (см. раздел "Параметры утилиты извлечения" на стр. 267) для обеспечения согласованности данных в извлеченной базе данных и на сервере. При запуске утилиты на третьем уровне изоляции возможно увеличение времени обратной передачи на сервере от других пользователей из-за большого количества требуемых блокировок. Рекомендуется запускать утилиту извлечения при низкой нагрузке на сервер, либо запускать ее из копии базы данных (см. раздел "Построение эффективной процедуры извлечения" на стр. 167).

Объекты,
принадлежащие **dbo**

Пользователь **dbo** является владельцем набором системных объектов в базе данных Adaptive Server Anywhere, совместимых с Adaptive Server Enterprise.

В случае Adaptive Server Anywhere утилита извлечения не выгружает объекты, созданные для пользователя **dbo**, при создании базы данных. Изменения, выполненные в отношении этих объектов, как, например, переопределение системной процедуры, при выгрузке данных теряются. Любые объекты, созданные с идентификатором пользователя **dbo**, начиная с инициализации базы данных, выгружаются утилитой извлечения, благодаря чему эти объекты сохраняются.

Извлечение удаленной базы данных в Sybase Central

При запуске из Sybase Central утилита извлечения выполняет следующие задачи, связанные с созданием и синхронизацией подписок SQL Remote:

- ◆ Создание командного файла для формирования удаленной базы данных, содержащей копию данных в указанных публикациях;
- ◆ Создание необходимых объектов SQL Remote, например, типов сообщений, идентификаторов издателя и удаленных пользователей, публикации и подписки для того, чтобы удаленная база данных могла получать сообщения от консолидированной базы данных и отправлять ей сообщения;
- ◆ Активизация подписки как в консолидированной, так и в удаленной базах данных.

❖ Процедура извлечения удаленной базы данных из запущенной базы данных

- 1 Выполните подключение к базе данных.
- 2 Щелкните правой кнопкой по базе данных и выберите во всплывающем меню пункт Extract Database.
- 3 Выполняйте указания мастера.

❖ Процедура извлечения удаленной базы данных из файла базы данных или запущенной базы данных

- 1 Откройте папку Utilities.
- 2 В правой области окна дважды щелкните по пункту Extract Database.
- 3 Выполняйте указания мастера.

Утилита извлечения

Назначение Извлечение удаленной базы данных Adaptive Server Anywhere из консолидированной базы данных Adaptive Server Enterprise или Adaptive Server Anywhere.

Синтаксис { **ssxtract** | **dbxtract** } [*параметры*] [*папка*] *подписчик*

Параметр	Описание
-an <i>база-данных</i>	Создание файла базы данных с теми же параметрами, что выгружаемая база данных, и его автоматическая перезагрузка
-ac " <i>ключевое-слово=значение; ...</i> "	Выполнение подключения к базе данных, указанной в строке подключения, для перезагрузки
-b	Запрет активизации подписок
-c " <i>ключевое-слово=значение; ...</i> "	Указание параметров подключения к базе данных
-d	Выгрузка только данных
-e <i>язык, кодовая-таблица</i>	Определение используемого языка
-f	Извлечение полностью определенных публикаций
-ii	Внутренняя выгрузка, внутренняя перезагрузка
-ix	Внутренняя выгрузка, внешняя перезагрузка
-j <i>счетчик</i>	Итерационный счетчик для операторов создания представлений
-l <i>уровень</i>	Выполнение всех операций извлечения на указанном уровне изоляции
-k	Закрытие окна по завершении
-n	Извлечение только определения схемы
-o <i>файл</i>	Вывод сообщений в файл
-p <i>символ</i>	Знак перехода
-q	Режим работы с минимальной нагрузкой: без печать сообщений и отображения окон
-r <i>файл</i>	Определяет имя созданного командного файла перезагрузки Interactive SQL (по умолчанию " <i>reload.sql</i> ")
-u	Неупорядоченные данные
-v	Расширенный вывод сообщений
-x	Использование загрузки внешних таблиц
-xf	Исключение внешних ключей

Параметр	Описание
-xi	Внешняя выгрузка, внутренняя перезагрузка
-xp	Исключение хранимых процедур
-xt	Исключение триггеров
-xv	Исключение представлений
-xx	Внешняя выгрузка, внешняя загрузка
-y	Перезапись командного файла без подтверждения
directory	Папка, в которую записываются файлы. Этого не требуется при использовании -an или -ac
subscriber	Подписчик, для которого должна быть извлечена база данных

Описание

ssxtract - утилита извлечения для Adaptive Server Enterprise. Утилита выполняется в Adaptive Server Enterprise и создает командный файл для удаленной базы данных Adaptive Server Anywhere.

dbxtract - утилита извлечения для Adaptive Server Anywhere. Она выполняется в базе данных Adaptive Server Anywhere и создает командный файл для удаленной базы данных Adaptive Server Anywhere.

Для создания удаленных баз данных Adaptive Server Enterprise утилит извлечения нет.

Утилитой извлечения создается командный файл и набор связанных файлов данных. Для создания объектов базы данных и загрузки данных в удаленную базу данных в недавно инициализированной базе данных Adaptive Server Anywhere может выполняться командный файл.

Имя командного файла по умолчанию - *reload.sql*.

Если удаленный пользователь имеет идентификатор пользователя группы, утилита извлечения извлекает все идентификаторы членов этой группы. При этом все пользователи в каждой удаленной базе данных могут использовать различные идентификаторы пользователей, что позволяет избежать необходимости реализации особых методов извлечения.

Примечания относительно SSXtract

Не для всех объектов Adaptive Server Enterprise имеются соответствующие объекты в Adaptive Server Anywhere. У утилиты *ssxtract* имеются следующие ограничения:

- ◆ **Одна база данных.** Все извлеченные объекты должны быть в одной базе данных Adaptive Server Enterprise.
- ◆ **Пароли.** Пароли для извлеченных идентификаторов пользователей совпадают с самими идентификаторами пользователей.
- ◆ **Полномочия.** Извлеченные идентификаторы пользователей имеют полномочия REMOTE DBA.
- ◆ **Поименованные ограничения.** Такие ограничения извлекаются как ограничения CHECK в Adaptive Server Anywhere.
- ◆ **Системные таблицы.** Набор системных таблиц Adaptive Server Anywhere создается в TEMPDB из системных таблиц Adaptive Server Enterprise процедурой *sp_populate_sql_anywhere* SQL Remote. Извлеченная схема поступает из этих временных системных таблиц.

☞ Для получения дополнительной информации о параметрах утилиты извлечения см. раздел "Параметры утилиты извлечения" на стр. 267.

Параметры утилиты извлечения

Создание базы данных для перезагрузки (-an). Использование этого параметра позволяет объединить операции выгрузки базы данных, создания новой базы данных и загрузки данных.

Например, следующая команда (вводится в одной строке), создает новый файл базы данных с именем *asacopy.db* и копирует в него схему и данные для подписчика *field_user asademo.db*:

```
dbxtract -c "uid=dba; pwd=sql; dbf=asademo.db" -an
asacopy.db field_user
```

При использовании этого параметра копия данных на диске не создается, поэтому в командной строке не требуется указывать папку для выгрузки. Это обеспечивает лучшую защиту для данных, но ценой некоторого уменьшения производительности.

Перезагрузка данных в существующую базу данных (-ac). Использование этого параметра позволяет объединить операции выгрузки базы данных и перезагрузки результатов в существующую базу данных.

Например, следующая команда (которая должна вводиться целиком в пределах одной строки) загружает копию данных для подписчика *field_user* в существующий файл базы данных с именем *newdemo.db*:

```
dbxtract -c "uid=dba; pwd=sql; dbf=asademo.db" -ac
"uid=dba; pwd=sql; dbf=newdemo.db" field_user
```

При использовании данного параметра копия данных на диске не создается, поэтому в командной строке не требуется указывать папку для выгрузки. Это обеспечивает лучшую защиту для данных, но ценой некоторого уменьшения производительности.

Запрет автоматической активизации подписок (-b). Если выбран этот параметр, для начала репликации необходимо активизировать подписки в консолидированной базе данных (для удаленной базы данных) и в удаленной базе данных (для консолидированной базы данных), явно используя оператор START SUBSCRIPTION.

Параметры подключения (-c). Набор параметров подключения в виде строки.

- ◆ **Параметры подключения dbxtract.** Пользователь должен иметь полномочия администратора БД для обеспечения наличия полномочий на работу со всеми таблицами в базе данных.

Например, нижеприведенный оператор (вводится в одной строке) извлекает базу данных для удаленного пользователя с идентификатором *joe_remote* из базы данных *asademo*, запущенной на сервере *sample_server*, при использовании идентификатора пользователя *dba* и пароля *sql*. Данные выгружаются в папку *c:\unload*.

```
ssxtract -c "eng=sample_server; dbn=sademo;
uid=dba; pwd=sql" c:\extract joe_remote
```

При отсутствии параметров подключения используются параметры подключения из переменной среды SQLCONNECT, если она задана.

- ◆ **Параметры подключения ssxtract.** Поддерживаются следующие параметры подключения:

Параметр	Описание
UID	Имя пользователя
PWD	Пароль
DBN	Имя базы данных (необязательный параметр) При отсутствии этого параметра подключение выполняется к базе данных по умолчанию для указанного имени пользователя.

Параметр	Описание
ENG	Имя Adaptive Server Enterprise.

ssxtract не может извлекать пароли. Поэтому при извлечении пароли совпадают с идентификаторами пользователей.

Выгрузка только данных (-d). При выборе данного параметра определение схемы не выгружается, а публикации и подписки в удаленной базе данных не создаются. Этот параметр используется в случаях, когда удаленная база данных уже существует, имеет надлежащую схему и должна быть только заполнена данными.

Использование указанного языка (-e). Этот параметр применим только к Adaptive Server Enterprise.

Задание информации о языке для Adaptive Server Enterprise. Строка языка имеет следующий формат:

"название_языка, кодовая_таблица [, порядок_сортировки]"

По умолчанию Message Agent использует значение языка, определенное в файле *sybase\locales\locales.dat*.

Если *название_языка* и *кодовая_таблица* не указаны, Message Agent получает их от Adaptive Server Enterprise. Если не указан *порядок_сортировки*, Message Agent использует двоичный порядок сортировки (по значениям байтов).

Извлечение полностью определенных публикаций (-f). В большинстве случаев для удаленных баз данных нет необходимости в извлечении полностью определенных публикаций, так как обычно это предполагает так или иначе репликацию всех строк назад в консолидированную базу данных.

Однако может возникнуть необходимость в полностью определенных публикациях в трехуровневых системах, либо для систем, где удаленная база данных имеет строки, которые отсутствуют в консолидированной базе данных.

Внутренняя выгрузка, внутренняя загрузка (-ii). Использование данного параметра вынуждает сценарий перезагрузки использовать для выгрузки и загрузки данных, соответственно, внутренние операторы UNLOAD и LOAD TABLE вместо операторов Interactive SQL OUTPUT и INPUT.

Данная комбинация операций является поведением по умолчанию.

Пути к файлам данных для внешних операций указываются относительно текущей рабочей папки *dbxtract*, в то время как для внутренних операторов - относительно сервера.

Внутренняя выгрузка, внешняя загрузка (-ix). Использование этого параметра вынуждает сценарий перезагрузки использовать внутренний оператор UNLOAD для выгрузки данных и оператор INPUT Interactive SQL, чтобы загрузить данные в новую базу данных.

Пути к файлам данных для внешних операций указываются относительно текущей рабочей папки *dbxtract*, в то время как для внутренних операторов - относительно сервера.

Итерационный счетчик для представлений (-j). Если в консолидированной базе данных имеются вложенные представления, этот параметр определяет максимальное число итераций для использования при извлечении представлений.

Выполнение извлечения на указанном уровне изоляции (-l). Установка по умолчанию - нулевой уровень изоляции. Производить извлечение базы данных из активного сервера необходимо на 3 уровне изоляции (см. раздел "Параметры утилиты извлечения" на стр. 267) для обеспечения совместимости данных в извлеченной базе данных с данными на сервере. Увеличение уровня изоляции может привести к большому числу блокировок, используемых утилитой извлечения, и может ограничить использование базы данных другими пользователями.

Выгрузка только определения схемы (-n). При данном определении никакие данные не выгружаются. Файл перезагрузки содержит операторы SQL для создания только структуры базы данных. Для загрузки данных по системе передачи сообщений можно использовать оператор SYNCHRONIZE SUBSCRIPTION. Публикации, подписки, полномочия PUBLISH и SUBSCRIBE являются частью схемы.

Вывод сообщений в файл (-o). Вывод сообщений процесса извлечения в файл для последующего просмотра.

Знак перехода (-r). Использование этого параметра позволяет заменить знак перехода, применяемый по умолчанию (\), на другой символ.

Режим работы с минимальной нагрузкой (-q). Сообщения не отображаются, за исключением сообщений об ошибках. Этот параметр из других сред не доступен. Он доступен только из утилиты командной строки.

Имя файла перезагрузки (-r). Имя по умолчанию для командного файла перезагрузки - *reload.sql* в текущей папке. Данный параметр позволяет определить для этого файла иное имя.

Вывод неупорядоченных данных (-u). По умолчанию данные в каждой таблице сортируются по первичному ключу. Выгрузка с параметром -u происходит быстрее, но загрузка данных в удаленную базу данных - медленнее.

Расширенный режим (-v). Отображение имен выгружаемых таблиц и количества выгруженных строк. При использовании оператора SELECT этот оператор также отображается.

Исключение определений внешних ключей (-xf). Данный параметр используется, если удаленная база данных содержит поднабор схемы консолидированной базы данных, а некоторые ссылки на внешние ключи в удаленной базе данных отсутствуют.

Внешняя выгрузка, внутренняя загрузка (-xi). Поведение по умолчанию при выгрузке базы данных заключается в использовании оператора UNLOAD, который выполняется сервером базы данных. Если выбрана внешняя выгрузка, *dbxtract* использует вместо этого оператор OUTPUT. Оператор OUTPUT выполняется в узле клиента.

Пути к файлам данных применительно к внешним операциям указывается относительно текущей рабочей папки *dbxtract*, в то время как применительно к внутренним операторам - относительно сервера.

Исключение хранимых процедур (-xp). Хранимые процедуры не извлекаются из базы данных.

Исключение триггеров (-xt). Триггеры не извлекаются из базы данных.

Исключение представлений (-xv). Представления не извлекаются из базы данных.

Внешняя выгрузка, внешняя загрузка (-xx). Использование оператора OUTPUT для выгрузки данных и оператора INPUT для загрузки данных в новую базу данных.

Поведение при выгрузке по умолчанию заключается в использовании оператора UNLOAD, а при загрузке - оператора LOAD TABLE. Внутренние операторы UNLOAD и LOAD TABLE выполняются быстрее, чем OUTPUT и INPUT.

Для внешних операций пути к файлам данных указываются относительно текущей рабочей папки *dbxtract*, в то время как для внутренних операторов - относительно сервера.

Выполнение операций без подтверждения (-y). Если этот параметр не установлен, выдаются запросы на подтверждение обновления существующего командного файла.

SQL Remote Open Server

Назначение Получение данных репликации с Replication Server и их обработка в очереди с сохранением SQL Remote. Данная утилита необходима только для баз данных, участвующих как в Replication Server (и использующих Replication Agent), так и в репликациях SQL Remote.

Синтаксис **ssqueue** [*параметры*] [*имя-open-server*]

Параметр	Описание
<i>имя-open-server</i>	Имя Open Server, которое должно быть объявлено в файле интерфейсов
-с " <i>ключевое-слово=значение; ...</i> "	Указание параметров подключения к базе данных
-сq " <i>ключевое-слово=значение; ...</i> "	Указание параметров подключения к базе данных для очереди с сохранением
-dl	Отображение сообщений в окне
-k	Закрытие окна по завершении
-o <i>файл</i>	Вывод сообщений в файл
-os <i>файл</i>	Максимальный размер файла для регистрации выводимых сообщений
-ot <i>файл</i>	Усечение файла и регистрация выводимых сообщений
-q	Выполнение в свернутом окне
-ud	Выполнение как демон [UNIX]
-v	Работа в расширенном режиме

Описание SQL Remote Open Server используется для обеспечения возможности базы данных Adaptive Server Enterprise участвовать в репликации SQL Remote и в то же время являться первичным узлом в системе Replication Server (либо узлом репликации с использованием асинхронных вызовов процедур).

Имена исполняемых файлов следующие:

- ◆ **ssqueue.exe** - операционные системы Windows;
- ◆ **ssqueue** - операционные системы UNIX.

Подробное описание параметров

имя-open-server Replication Server должен выполнить подключение к SQL Remote Open Server, который при этом должен иметь имя, указываемое в данном параметре. Это имя задается в командной строке и должно соответствовать хосту и записи запроса в файле интерфейсов на компьютере, на котором запущен SQL Remote Open Server, а также записи запроса в файле интерфейсов на компьютере, на котором запущен Replication Server.

Файл интерфейсов имеет имя *sql.ini* в операционных системах Windows и *interfaces* - в UNIX.

Значение по умолчанию для имени Open Server - **SSQueue**.

-с Определение параметров подключения к базе данных, содержащей реплицируемые данные. Данное подключение требуется для SQL Remote Open Server для получения доступа к системным таблицам SQL Remote.

Параметры подключения должны исходить из следующего списка:

Параметр	Описание
UID	Имя пользователя
PWD	Пароль
DBN	Имя базы данных (необязательный параметр) При отсутствии этого параметра подключение выполняется к базе данных по умолчанию для указанного имени пользователя.
ENG	Имя сервера

-sq Определение параметров подключения для очереди с сохранением. Если этот параметр не указан, значения устанавливаются по умолчанию как значения, указанные в `-c`.

-dl Отображение сообщений в окне Message Agent или в командной строке, а также в файле журнала.

-k Закрытие окна по завершении.

-o Добавление выводимой информации в конец файла журнала. По умолчанию информация выводится на экран.

-os Определение максимального размера файла для регистрации выводимых сообщений. Разрешенный объем может быть определен как *n* (в байтах), *nK* (Кб) или *nM* (Мб). По умолчанию никаких ограничений нет; нижний предел составляет 10000 байтов.

Текущий размер файла проверяется перед выводом в файл сообщений журнала SQL Remote. Если сообщение их журнала приведет к превышению размера файла указанного размера, то SQL Remote переименовывает файл вывода следующим образом: *ggmmddxx.dbr* (для dbremote) или *ggmmddxx.ssr* (для ssremote), где *xx* - последовательные символы в пределах от AA до ZZ, а *ggmmdd* соответствуют текущему году, месяцу и дню.

Этот параметр позволяет вручную удалять старые файлы журналов и освобождать дисковое пространство, если Message Agent работает непрерывно в течение долгого времени.

-ot Усечение файла журнала с последующим добавлением выводимых сообщений в его конец. По умолчанию в таких случаях информация выводится на экран.

-q Только для оконных операционных систем: выполнение Message Agent в свернутом окне.

-ud Указание параметра `-ud` на платформах UNIX позволяет запустить SQL Remote Open Server как демон.

При выполнении демона необходимо также указать параметр `-o` или `-ot` для записи в выводимой информации журнала.

-v Расширенный вывод. Этот параметр отображает на экране операторы SQL, содержащиеся в сообщениях, и, если указаны параметры `-o` или `-ot`, в файле журнала.

Параметры SQL Remote

Назначение Параметры репликации - параметры базы данных, используемые для управления поведением системы репликации.

Синтаксис Anywhere **SET [TEMPORARY] OPTION**
 ... [*идентификатор-пользователя.* | **PUBLIC.**]*имя_параметра* = [значение_параметра]

Синтаксис Enterprise **exec sp_remote_option** *имя-параметра*, *значение-параметра*

Параметры	Аргумент	Описание
	<i>имя_параметра</i>	Имя изменяемого параметра
	<i>значение-параметра</i>	Строка, содержащая установку для параметра

Описание Доступны следующие параметры:

ПАРАМЕТРЫ	ЗНАЧЕНИЯ	ЗНАЧЕНИЯ ПО УМОЛЧАНИЮ
Blob_threshold	целое число, в Кб	256
Compression	от -1 до 9	6
Delete_old_logs	ON, OFF	OFF
External_remote_options	ON, OFF	OFF
Qualify_owners	ON, OFF	OFF
Quote_all_identifiers	ON, OFF	OFF
Replication_error	<i>имя процедуры</i>	NULL
Save_remote_passwords	ON, OFF	ON
SR_Date_Format	<i>строка-даты</i>	yyyy/mm/dd
SR_Time_Format	<i>строка-времени</i>	hh:nn:ss.Ssssss
SR_Timestamp_Format	<i>строка-метки-времени</i>	yyyy/mm/dd hh:nn:ss.Ssssss
Subscribe_by_remote	ON,OFF	ON
Verify_threshold	<i>integer</i>	256
Verify_all_columns	ON,OFF	OFF

Эти параметры используются Message Agent и должны быть заданы для *идентификатора* пользователя, указываемого в команде Message Agent. Использование параметров также может быть открытым и совместным.

Ниже приведено описание этих параметров:

Параметр Blob_threshold. Любое значение, имеющее длину больше, чем параметр Blob_threshold, реплицируется как blob-объект. Это означает, что объект разделяется на части и реплицируется по фрагментам, а затем воссоздается путем объединения частей на узле получателя при использовании переменной SQL.

Если blob-объекты реплицируются в системе репликации с Adaptive Server Enterprise, необходимо обеспечить установку большего значения Blob_threshold, чем длина наибольшего реплицируемого blob-объекта.

☞ Для получения информации относительно репликации blob-объектов и Adaptive Server Enterprise см. раздел "Репликация blob-объектов" на стр. 71.

Параметр Compression. Задает уровень сжатия для сообщений. Устанавливаемые значения находятся в диапазоне от -1 до 9 и имеют следующий смысл:

- ◆ -1 Отправка сообщений в формате версии 5. Message Agent (*dbremote* или *ssremote*) предыдущих версий SQL Remote не сможет читать сообщения, отправленные в формате версии 6. До обновления всех Message Agent системы до версии 5 необходимо обеспечить, чтобы для параметра COMPRESSION было установлено значение -1.
- ◆ 0 Сжатие не производится.
- ◆ От 1 до 9 Увеличение степени сжатия. Процедура создания сообщений с высокой степенью сжатия может длиться дольше, чем создание сообщений с низким сжатием.

Параметр Delete_old_logs. Этот параметр используется SQL Remote и Replication Agent в Adaptive Server Anywhere. Установка по умолчанию: OFF. Если для этого параметра установлено значение ON, Message Agent (DBREMOTE) удаляет каждый старый журнал транзакций, если все изменения, которые он содержит, уже отправлены, и подтверждено их получение.

External_remote_options. Данный параметр используется SQL Remote для указания того, должны ли параметры системы передачи сообщений быть сохранены в базе данных (OFF) или на внешних носителях (ON). По умолчанию этот параметр установлен в OFF.

Параметр Qualify_owners. Используется для указания того, должны ли операторы SQL, реплицируемые SQL Remote, использовать определенные имена объектов. Значение по умолчанию для Adaptive Server Anywhere - ON, а для Adaptive Server Enterprise - OFF.

Определение владельцев в установках Adaptive Server Enterprise требуется редко, так как владельцем объектов является **dbo**. При отсутствии необходимости определения в системах Adaptive Server Anywhere сообщения будут немного меньше по объему, если для этого параметра установлено значение OFF.

Параметр Quote_all_identifiers. Используется для указания того, должны ли операторы SQL, реплицируемые SQL Remote, использовать цитированные идентификаторы. Значение параметра по умолчанию - OFF.

Если значение этого параметра - OFF, *dbremote* цитирует идентификаторы, которые требуют кавычек в Adaptive Server Anywhere (это выполняется всегда), а *ssremote* их не цитирует. Если значение этого параметра - ON, цитируются все идентификаторы.

Параметр Replication_error. Определяет хранимую процедуру, вызываемую Message Agent при возникновении ошибки SQL. По умолчанию никакие процедуры не вызываются.

Процедура обработки ошибок репликации должна иметь единственный аргумент типа CHAR, VARCHAR или LONG VARCHAR. Данная процедура вызывается один раз при получении сообщения об ошибке SQL и один раз при выполнении оператора SQL, вызывающего ошибку.

Несмотря на то, что данный параметр позволяет отслеживать ошибки и контролировать их появление при репликации, возможность возникновения ошибок необходимо минимизировать на этапе настройки системы, поскольку данный параметр не обеспечивает устранения таких ошибок.

Можно использовать таблицу с DEFAULT CURRENT REMOTE USER для записи удаленных узлов, вызывающих ошибки.

Параметр Save_remote_passwords. При вводе пароля в диалоговое окно системы передачи сообщений при первом подключении эти значения сохраняются. По умолчанию параметр Save_remote_passwords установлен в значение ON, т. е. пароли сохраняются. При внешнем сохранении параметров

системы передачи сообщений (вместо сохранения этих параметров в базе данных) может понадобиться не сохранять пароли. Можно запретить сохранение паролей путем установки этого параметра в NO.

Параметр SR_Date_Format. Message Agent использует этот параметр при репликации столбцов, хранящих даты. Параметр является строкой, построенной из следующих символов:

Символ	Описание
yy	Год (две цифры)
yyyy	Год (4 цифры)
mm	Месяц (2 цифры)
mmm	Символьный формат месяца
dd	День (2 цифры)

При репликации происходит замена символов реплицируемой датой.

При задании символьного формата **mm** в верхнем регистре соответствующие символы также являются символами верхнего регистра.

В случае числовых форматов регистр установки параметра управляет заполнением нулями. Если символы одного регистра (например, DD), число заполняется нулями. Если символы разного регистра (например, Mm), число не заполняется нулями.

Параметр SR_Time_Format. Message Agent использует этот параметр при репликации столбцов, хранящих значения времени. Параметр является строкой, построенной из следующих символов:

Символ	Описание
hh	Часы (2 цифры, 24-часовой формат времени)
nn	Минуты (2 цифры)
mm	Минуты (2 цифры), если перед ними указано двоеточие (как в hh:mm)
ss[s ...]	Секунды (2 цифры) плюс (необязательно) доли секунды.

Использование смешанного регистра в строке форматирования отменяет начальные нули.

SR_Timestamp_Format. С помощью этого параметра Message Agent производит репликацию информации о дате и времени. В случае Adaptive Server Anywhere это типы данных timestamp, datetime и smalldatetime. В случае Adaptive Server Enterprise это типы данных datetime и smalldatetime.

Строки формата берутся из значений SR_Date_Format и SR_Time_Format.

Установка по умолчанию - SR_Date_Format, после которой следует значение SR_Time_Format.

Параметр Subscribe_by_remote. При установке для данного параметра значения ON для операций удаленных баз данных со строками с подпиской по значению, являющемуся NULL или пустой строкой, предполагается, что на строку подписан удаленный пользователь. При установке для данного параметра значения OFF предполагается, что удаленный пользователь не подписан на строку.

Единственное ограничение данного параметра заключается в том, что при выполнении операций INSERT (или UPDATE) удаленным пользователем для строк с действительно пустыми или имеющими значение NULL выражениями подписки возникнут ошибки (для информации, хранимой только в консолидированной базе данных). Причина ошибок очевидна и может быть

устранена путем назначения в систем репликации значения подписки, не принадлежащей тому ли иному удаленному пользователю.

☞ Для получения дополнительной информации об этом параметре см. разделы "Использование параметра `Subscribe_by_remote` со связями "многие ко многим" на стр. 118 и на стр. 165.

Параметр `Verify_threshold`. Если тип данных столбца имеет большую длину, чем пороговое значение, при репликации `UPDATE` старые значения для столбца не проверяются. Значением по умолчанию является уровень 1000.

Данный параметр служит для поддержания небольшого размера сообщений `SQL Remote`, но имеет недостаток, заключающийся в том, конфликты обновлений длинных значений не обнаруживаются.

Параметр `Verify_all_columns`. Установка по умолчанию – `OFF`. При установленном значении `ON` сообщения, содержащие обновления, изданные локальной базой данных, отправляются, включая все значения столбцов, и конфликт в любом столбце приводит к запуску триггера `RESOLVE UPDATE` в базе данных подписчика.

Утилита извлечения для `Adaptive Server Enterprise` так устанавливает общедоступный параметр в удаленных базах данных `Adaptive Server Anywhere`, чтобы он соответствовал установке в базе данных `Adaptive Server Enterprise`.

Примеры

- ◆ Нижеприведенный оператор устанавливает значение `OFF` для параметра `Verify_all_columns` в `Adaptive Server Anywhere` для всех пользователей:

```
SET OPTION PUBLIC.Verify_all_columns = 'OFF'
```

- ◆ Нижеприведенные операторы устанавливают значение `OFF` для параметра `Verify_all_columns` в `Adaptive Server Enterprise`:

```
exec sp_remote_option Verify_all_columns, 'OFF' go
```

В `Adaptive Server Enterprise` параметры репликации используются только `SQL Remote`.

Процедуры перехватчиков событий SQL Remote

Хранимые процедуры со следующими именами и аргументами обеспечивают интерфейс для настройки синхронизации в базах данных SQL Remote.

Примечания

Если не указано иное, последующее относится к процедурам перехватчиков событий:

- ◆ Хранимым процедурам должны быть назначены либо полномочия DBA (Adaptive Server Anywhere), либо полномочия dbo (Adaptive Server Enterprise).
- ◆ Процедуры не должны подтверждать операции или выполнять откат операций, а также осуществлять любые действия с неявным подтверждением. Действия процедур автоматически подтверждаются вызывающими приложениями.
- ◆ Можно использовать перехватчики событий для поиска и обнаружения неполадок, также в расширенном режиме Message Agent.
- ◆ Перехватчики событий для *dbremote* и *ssremote* отличаются только именами.

Таблица #hook_dict

Таблица *#hook_dict* создается следующим оператором CREATE непосредственно перед вызовом перехватчика событий:

```
CREATE table #hook_dict (
  name VARCHAR(128) NOT NULL UNIQUE,
  value VARCHAR(255) NOT NULL )
```

Message Agent использует таблицу *#hook_dict* для передачи значений функциям перехвата; подключаемые функции используют таблицу *#hook_dict* для передачи значений назад к Message Agent.

sp_hook_dbremote_begin и sp_hook_ssrmt_begin

Функция

Данная хранимая процедура используется для добавления специальных действий в начало процесса при репликации.

Строки в таблице #hook_dict

Имя	Значения	Описание
send	true или false	Информирует о том, выполняет ли процесс при репликации операцию отправки.
receive	true или false	Информирует о том, выполняет ли процесс при репликации операцию получения/

Описание

Если процедура с таким именем существует, она вызывается при запуске Message Agent.

sp_hook_dbremote_end и sp_hook_ssrmt_end

Функция

Данная хранимая процедура используется для добавления специальных действий непосредственно перед закрытием Message Agent.

Строки таблицы #hook_dict

Имя	Значения	Описание
send	true или false	Информирует о том, выполняет ли процесс репликации операцию отправки.
receive	true или false	Информирует о том, выполняет ли процесс репликации операцию получения.
exit code	integer	Ненулевой код завершения указывает на ошибку.

Описание Если процедура с этим именем существует, она вызывается как последнее событие перед закрытием Message Agent.

sp_hook_dbremote_shutdown и sp_hook_ssrm_shutdown

Функция Данная хранимая процедура используется для инициирования завершения работы Message Agent.

Строки таблицы #hook_dict	Имя	Значения	Описание
	send	true или false	Информирует о том, выполняет ли процесс репликации операцию отправки.
	receive	true или false	Информирует о том, выполняется ли процессом репликации операция получения.
	shutdown	true или false	При вызове процедуры данная строка имеет значение false . Если процедура изменяет строку в значение true , Message Agent завершает работу.

Описание Если процедура с этим именем существует, она вызывается, когда Message Agent не производит ни отправки, ни получение сообщений, и разрешает иницируемое перехватчиком событий завершение работы Message Agent.

sp_hook_dbremote_receive_begin и sp_hook_ssrm_receive_begin

Функция Данная хранимая процедура используется для выполнения действий перед началом этапа получения сообщений при репликации.

Строки в #hook_dict Нет.

sp_hook_dbremote_receive_end и sp_hook_ssrm_receive_end

Функция Данная хранимая процедура используется для выполнения действий после окончания этапа получения сообщений при репликации.

Строки в #hook_dict Нет.

sp_hook_dbremote_send_begin и sp_hook_ssrm_send_begin

Функция Данная хранимая процедура используется для выполнения действий перед началом этапа отправки сообщений при репликации.

Строки в #hook_dict Нет.

sp_hook_dbremote_send_end и sp_hook_ssrm_send_end

Функция Данная хранимая процедура используется для выполнения действий после окончания этапа отправки сообщений при репликации.

Строки в #hook_dict Нет.

sp_hook_dbremote_message_sent и sp_hook_ssrm_message_sent

Функция Данная хранимая процедура используется для выполнения действий после отправки любого сообщения.

Строки в #hook_dict	Имя	Значения
	remote user	Адресат сообщения

sp_hook_dbremote_message_missing и sp_hook_ssrm_message_missing

Функция Данная хранимая процедура используется для выполнения действий тогда, когда Message Agent принял решение об отсутствии одного или большего количества сообщений от удаленного пользователя.

Строки в #hook_dict	Имя	Значения
	remote user	Имя удаленного пользователя, который должен выполнить повторную передачу сообщений.

sp_hook_dbremote_apply_begin и sp_hook_ssrm_apply_begin

Функция Данная хранимая процедура используется для выполнения действий непосредственно перед обработкой Message Agent набора сообщений от пользователя.

Строки в #hook_dict	Имя	Значения
	remote user	Имя удаленного пользователя, отправившего сообщения, которые будут обрабатываться.

sp_hook_dbremote_apply_end и sp_hook_ssrm_apply_end

Функция Используйте эту хранимую процедуру для выполнения действий непосредственно после того, как Message Agent обработал набор сообщений от пользователя.

Строки в #hook_dict	Имя	Значения
	remote user	Имя удаленного пользователя, отправившего сообщения, которые были обработаны.

Системные объекты для Adaptive Server Anywhere

Об этой главе

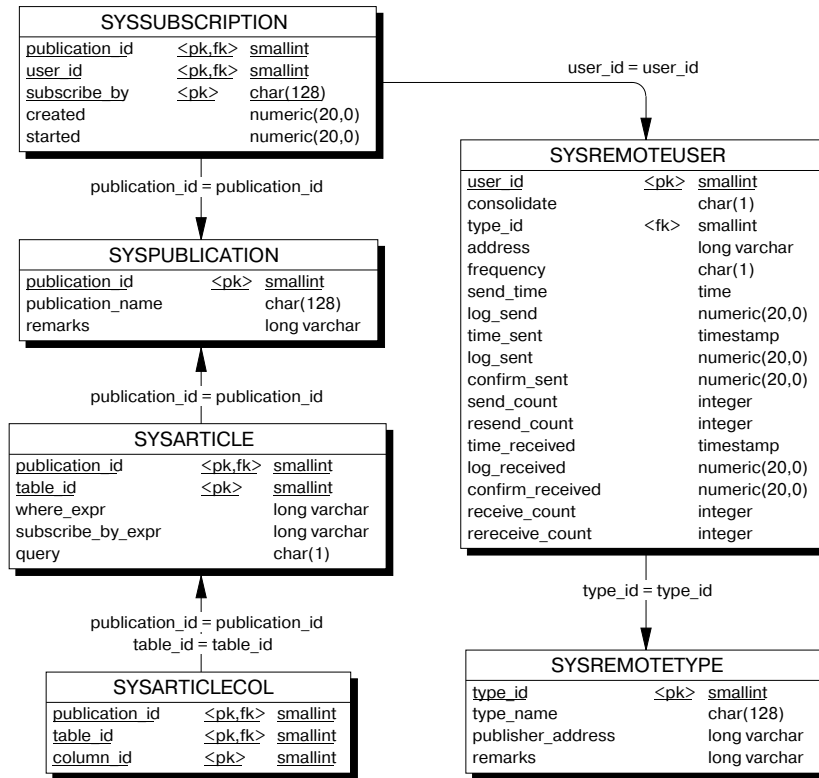
Системная информация SQL Remote находится в папке Adaptive Server Anywhere. В наборе системных представлений эта информация представлена в более удобном для прочтения виде.

Содержание

Раздел	Страница
Системные таблицы SQL Remote	280
Системные представления SQL Remote	285

Системные таблицы SQL Remote

В данном разделе описаны системные таблицы, используемые SQL Remote для определения и управления данными SQL Remote. В приведенной ниже схеме стрелки обозначают связи между таблицами по внешнему ключу: стрелка направлена от внешней таблицы к первичной таблице.



В последующих разделах представлено более подробное описание этих таблиц.

Таблица SYSARTICLE

Назначение

Каждая строка описывает статью в публикации SQL Remote.

Столбцы

Столбец	Тип данных	Описание
publication_id	UNSIGNED INT	Публикация, частью которой является данная статья.
table_id	UNSIGNED INT	Каждая статья состоит из столбцов и строк из одной таблицы. Данный столбец содержит идентификатор этой таблицы.
where_expr	LONG VARCHAR	Данный столбец содержит условие поиска для статей, которые содержат поднабор строк, определенных разделом WHERE.
subscribe_by_expr	LONG VARCHAR	Данный столбец содержит выражение для статей, которые содержат поднабор строк, определенных выражением SUBSCRIBE BY.
query	CHAR(1)	Выражение SUBSCRIBE BY может

Столбцы	Столбец	Тип данных	Описание
			быть подзапросом, возвращающим множественные значения. Данный столбец содержит Y или N для указания того, является ли выражение подзапросом (Y) или нет (N). Этот столбец используется утилитой извлечения.

Таблица SYSARTICLECOL

Назначение Каждая строка идентифицирует столбец в статье, а именно, данный столбец, таблицу, в которой он находится, и публикацию, частью которой он является.

Столбцы	Столбец	Тип данных	Описание
	publication_id	UNSIGNED INT	Уникальный идентификатор публикации, частью которой является столбец.
	table_id	UNSIGNED INT	Таблица, которой принадлежит столбец.
	column_id	UNSIGNED INT	Идентификатор столбца из системной таблицы SYSCOLUMN.

Таблица SYSPUBLICATION

Назначение Каждая строка описывает публикацию SQL Remote.

Столбцы	Столбец	Тип данных	Описание
	publication_id	UNSIGNED INT	Уникальный идентификатор публикации.
	creator	UNSIGNED INT	Идентификатор пользователя, являющегося владельцем публикации.
	publication_name	VARCHAR (128)	Имя публикации.
	remarks	LONG VARCHAR	Комментарии.

Таблица SYSREMOTEOPTION

Назначение Каждая строка описывает значения параметра системы передачи сообщений SQL Remote.

Столбцы	Столбец	Тип данных	Описание
	option_id	UNSIGNED INT	Идентификатор параметра системы передачи сообщений.
	user_id	UNSIGNED INT	Идентификатор пользователя, для которого задан этот параметр.
	"setting00"	VARCHAR(255)	Значение параметра системы передачи сообщений.

Таблица SYSREMOTEOPTIONTYPE

Назначение Каждая строка описывает один из параметров системы передачи сообщений SQL Remote.

Столбец	Тип данных	Описание
option_id	UNSIGNED INT	Идентификатор параметра системы передачи сообщений.
type_id	SMALLINT	Идентификатор типа сообщений, использующего этот параметр
"option"	VARCHAR(128)	Имя параметра системы передачи сообщений.

Таблица SYSREMOTETYPE

Назначение Каждая строка описывает один из типов сообщений SQL Remote, включая адрес издателя.

Столбец	Тип данных	Описание
type_id	SMALLINT	Идентификатор типа сообщений.
type_name	VARCHAR(128)	Тип сообщений. Для каждого из указанных ниже параметров имеется отдельная строка: <ul style="list-style-type: none"> ◆ FILE ◆ MAPI ◆ VIM ◆ SMTP
publisher_address	VARCHAR(128)	Адрес издателя для типа сообщений type_name . SQL Remote получает сообщения с этого адреса.
remarks	LONG VARCHAR	Комментарии

Таблица SYSREMOTEUSER

Назначение Каждая строка описывает идентификатор пользователя с полномочиями REMOTE (подписчика) вместе с состоянием сообщений SQL Remote, отправленных этому пользователю и полученных от него.

Столбец	Тип данных	Описание
user_id	UNSIGNED INT	Идентификатор пользователя, имеющего полномочия REMOTE.
consolidate	CHAR(1)	Столбец содержит либо N, что указывает на наличие предоставленных пользователю полномочий REMOTE, либо Y, что указывает на наличие предоставленных пользователю полномочий CONSOLIDATE.
type_id	SMALLINT	Идентификатор системы передачи сообщений, которая использовалась для отправки

Столбцы	Столбец	Тип данных	Описание
			сообщений этому пользователю.
	address	LONG VARCHAR	Адрес, на который должны отправляться сообщения SQL Remote. Адрес должен соответствовать address_type .
	frequency	CHAR(1)	Периодичность отправки сообщений SQL Remote. "P" означает "периодически", "A" – "иногда".
	send_time	TIME	Указание времени следующей отправки сообщений данному пользователю.
	log_send	NUMERIC(20, 0)	Если log_send больше, чем log_sent , Message Agent осуществляет повторную передачу сообщений подписчику сразу при следующем запуске Message Agent.
	time_sent	TIMESTAMP	Время отправки последнего сообщения данному подписчику.
	log_sent	NUMERIC(20, 0)	Смещение в локальном журнале, соответствующее последней операции отправки данному подписчику.
	confirm_sent	NUMERIC(20, 0)	Смещение в журнале, соответствующее последней подтвержденной данным подписчиком операции.
	send_count	INT	Количество сообщений, отправленных SQL Remote данному подписчику.
	resend_count	INT	Счетчик, обеспечивающий однократную обработку сообщений в базе данных подписчика.
	time_received	DATETIME	Время получения последнего сообщения от данного подписчика.
	log_received	NUMERIC(20, 0)	Смещение в журнале базы данных подписчика, соответствующее операции, полученной в текущей базе данных последней.
	confirm_received	NUMERIC(20, 0)	Смещение в журнале базы данных подписчика, соответствующее операции, подтверждение которой было отправлено последним.
	receive_count	INT	Количество сообщений, полученных от данного подписчика.
	rereceive_count	INT	Счетчик, обеспечивающий однократную обработку сообщений в текущей базе данных.

Таблица SYSSUBSCRIPTION

Назначение

Каждая строка описывает подписку для одного идентификатора пользователя (пользователь должен иметь полномочия REMOTE) на одну публикацию.

Столбцы

Столбец	Тип данных	Описание
publication_id	UNSIGNED INT	Идентификатор публикации, на которую подписан пользователь.
user_id	UNSIGNED INT	Идентификатор пользователя, подписанного на публикацию.
subscribe_by	VARCHAR(128)	Этот столбец содержит значение соответствия для данной подписки для публикаций с выражением SUBSCRIBE BY.
created	NUMERIC(20, 0)	Смещение в журнале транзакций, при котором подписка была создана.
started	NUMERIC(20, 0)	Смещение в журнале транзакций, при котором подписка была активизирована.

Системные представления SQL Remote

Данный раздел описывает представления базы данных, используемые в SQL Remote для вывода и организации выводимой информации.

Представление SYSARTICLES

Назначение Каждая строка описывает статью.

Столбцы	Столбец	Описание
	publication_name	Публикация, частью которой является данная статья.
	table_name	Каждая статья состоит из столбцов и строк из одной таблицы. Данный столбец содержит имя этой таблицы.
	where_expr	Этот столбец содержит условие поиска для статей, которые содержат поднабор строк, определенных разделом WHERE.
	subscribe_by_expr	Этот столбец содержит выражение для статей, которые содержат поднабор строк, определенных выражением SUBSCRIBE BY.

Представление SYSARTICLECOLS

Назначение Каждая строка описывает столбец, который появляется в статье.

Столбцы	Столбец	Описание
	publication_name	Имя публикации, частью которой является столбец.
	table_name	Имя таблицы, которой принадлежит столбец.
	column_name	Имя столбца.

Представление SYSPUBLICATIONS

Назначение Перечисление имен всех публикаций.

Столбцы	Столбец	Описание
	publication_name	Имя публикации.
	creator	Владелец публикации.
	remarks	Комментарии.

Представление SYSREMOTEOPTIONS

Назначение Перечисление в более удобной для прочтения форме параметров системы передачи сообщений SQL Remote и их значений, хранимых в системных таблицах *SYSREMOTEOPTION* и *SYSREMOTEOPTIONTYPE*.

Столбцы	Столбец	Описание
	type_name	Тип системы передачи сообщений.
	"option"	Имя параметра.
	setting	Значение параметра.

Представление SYSREMOTEUSERS

Назначение

Вывод информации об удаленных пользователях и их статусе.

Столбцы

Столбец	Описание
user_name	Идентификатор пользователя с полномочиями REMOTE.
consolidate	Столбец содержит либо N, что указывает на наличие предоставленных пользователю полномочий REMOTE, либо Y, что указывает на наличие предоставленных пользователю полномочий CONSOLIDATE.
type_name	Имя типа сообщений, используемого для отправки сообщений этому пользователю.
address	Адрес, на который должны отправляться сообщения SQL Remote. Адрес должен соответствовать address_type .
frequency	Периодичность отправки сообщений SQL Remote.
send_time	Указание времени следующей отправки сообщений данному пользователю.
next_send	Указание времени следующей отправки сообщений данному пользователю в более удобной для прочтения форме.
log_send	Сообщения отправляются только тем подписчикам, для которых log_send больше, чем log_sent .
time_sent	Время отправки последнего сообщения данному подписчику.
log_sent	Смещение в локальном журнале, соответствующее последней операции отправки данному подписчику.
confirm_sent	Смещение в журнале, соответствующее последней подтвержденной данным подписчиком операции.
send_count	Количество сообщений, отправленных SQL Remote.
resend_count	Счетчик, обеспечивающий однократную обработку сообщений в базе данных подписчика.
time_received	Время получения последнего сообщения от данного подписчика.
log_received	Смещение в журнале базы данных подписчика, соответствующее операции, полученной в текущей базе данных последней.
confirm_received	Смещение в журнале базы данных подписчика, соответствующее операции, подтверждение которой было отправлено последним.
receive_count	Количество полученных сообщений.
rereceive_count	Счетчик, обеспечивающий однократную обработку сообщений в текущей базе данных.

Представление SYSSUBSCRIPTIONS

Назначение

Каждая строка представляет информацию о подписке.

Столбцы

Столбец	Описание
publication_name	Имя публикации, на которую подписан пользователь.
user_name	Идентификатор пользователя, подписанного на публикацию.
subscribe_by	Этот столбец содержит значение соответствия для данной подписки для публикаций с выражением SUBSCRIBE BY.
created	Смещение в журнале транзакций, при котором подписка была создана.
started	Смещение в журнале транзакций, при котором подписка была активизирована.

Системные объекты для Adaptive Server Enterprise

Об этой главе

Системная информация SQL Remote находится в наборе таблиц, которые называются системными таблицами SQL Remote. В наборе представлений, которые называются системными представлениями SQL Remote, эта информация представлена в более удобном для прочтения виде.

Содержание

Раздел	Страница
Системные таблицы SQL Remote	290
Системные представления SQL Remote	296
Таблицы очереди с сохранением	300

Системные таблицы SQL Remote

В данном разделе описаны системные таблицы, используемые SQL Remote для определения и управления данными SQL Remote.

Предостережение

Эти таблицы предназначены только для использования SQL Remote. Нельзя непосредственно изменять эти таблицы или их содержание.

Таблица #remote

Назначение

Эта временная таблица создается Message Agent для хранения имен текущего удаленного пользователя и текущего издателя. Данная таблица существует только в Adaptive Server Enterprise.

Столбцы

Столбец	Тип данных	Описание
current_remote_user	VARCHAR(128)	Текущий удаленный пользователь (из командной строки Message Agent).
current_publisher	VARCHAR(128)	Текущий издатель

Описание

Эта таблица не является системной таблицей. Таблица #remote содержит идентификаторы текущего удаленного пользователя и текущего издателя, используемые при подключении Message Agent для Adaptive Server Enterprise к серверу. Эта временная таблица находится в базе данных TEMPDB.

Значения из таблицы #remote могут использоваться в процедурах разрешения конфликтов.

Оператор CREATE TABLE для данной таблицы:

```
CREATE TABLE #remote (
    current_remote_user varchar(128),
    current_publisher varhcar(128)
)
```

Таблица имеет одну строку.

Таблица sr_article

Назначение

Каждая строка описывает статью в публикации SQL Remote.

Столбцы

Столбец	Тип данных	Описание
publication_id	INT	Публикация, частью которой является данная статья.
table_id	INT	Каждая статья состоит из столбцов и строк одной таблицы. Данный столбец содержит идентификатор этой таблицы.
where_expr	VARCHAR(128)	Данный столбец содержит условие поиска для статей, которые содержат поднабор строк, определенных разделом WHERE.
subscribe_by_expr	VARCHAR(128)	Данный столбец содержит выражение для статей, которые содержат поднабор строк, определенных

Столбцы	Столбец	Тип данных	Описание
	subscribe_by_view	VARCHAR(128)	выражением SUBSCRIBE BY. Этот столбец содержит имя представления для статей, которые содержат поднабор строк, определенных представлением.

Таблица sr_articlecol

Назначение Каждая строка идентифицирует столбец в статье, а именно, данный столбец, таблицу, в которой он находится, и публикацию, частью которой он является.

Столбцы	Столбец	Тип данных	Описание
	publication_id	INT	Уникальный идентификатор публикации, частью которой является столбец.
	table_id	INT	Таблица, которой принадлежит столбец.
	column_id	INT	Идентификатор столбца из системной таблицы SYSCOLUMN.

Таблица sr_marker

Назначение Обеспечение отправки сообщений, полученных Message Agent, удаленным базам данных в том же сеансе.

Столбцы	Столбец	Тип данных	Описание
	marker	DATETIME	Время обработки самых последних сообщений.

Описание Если консолидированная база данных использует два Message Agent, один для заполнения очереди с сохранением (-i), а другой - для получения и отправки сообщений (-r -s), то для обеспечения отправки полученных и обработанных в базе данных сообщений до закрытия Message Agent используется одна строка таблицы **sr_marker**.

Таблица sr_object

Назначение Содержит список объектов SQL Remote. Утилита извлечения не должна извлекать системные объекты SQL Remote. Процедура **sp_populate_sql_anywhere**, создающая набор системных таблиц Adaptive Server Anywhere в TEMPDB, получает список объектов SQL Remote из таблицы **sr_object**.

Столбцы	Столбец	Тип данных	Описание
	name	VARCHAR(128)	Имя объекта.
	type	CHAR(1)	Один из следующих типов объектов: <ul style="list-style-type: none"> ◆ U определяемая пользователем таблица ◆ V Представление ◆ P Процедура

Таблица sr_option

Назначение Каждая строка описывает параметр репликации, используемый SQL Remote.

Столбцы	Столбец	Тип данных	Описание
	option	VARCHAR(128)	Имя параметра.
	value	VARCHAR(128)	Значение данного параметра.


Описание  Для получения информации о доступных параметрах см. раздел "Параметры SQL Remote" на стр. 272.

Таблица sr_passthrough

Назначение Каждая строка описывает операцию ретрансляции, отправляемую пользователю или подписчикам на публикацию.

Столбцы	Столбец	Тип данных	Описание
	operation	VARCHAR(20)	Операция ретрансляции или часть операции ретрансляции, которая вводится с помощью sp_passthrough или sp_passthrough_piece .
	value	VARCHAR(255)	Значение столбца подписки, указывающее, какие пользователи должны получать данную операцию.
	id	INT	Пользователь, который должен получать операцию.

Таблица sr_publication

Назначение Каждая строка описывает публикацию SQL Remote.

Столбцы	Столбец	Тип данных	Описание
	publication_id	INT	Идентификатор публикации.
	publication_name	VARCHAR(128)	Имя публикации.

Таблица sr_publisher

Назначение Строка содержит идентификатор издателя.

Столбцы	Столбец	Тип данных	Описание
	user_id	INT	Идентификатор издателя.

Таблица sr_remotoption

Назначение Каждая строка описывает значения параметра системы передачи сообщений SQL Remote.

Столбцы	Столбец	Тип данных	Описание
	option_id	INTEGER	Идентификатор параметра системы передачи сообщений.
	user_id	INTEGER	Идентификатор пользователя, для которого задан этот параметр.
	"setting00"	VARCHAR(255)	Значение параметра системы

Столбцы	Столбец	Тип данных	Описание
			передачи сообщений.

Таблица sr_remoteoptiontype

Назначение Каждая строка описывает один из параметров системы передачи сообщений SQL Remote.

Столбцы	Столбец	Тип данных	Описание
	option_id	INTEGER	Идентификатор параметра системы передачи сообщений.
	type_id	INTEGER	Идентификатор типа сообщений, использующего этот параметр
	"option"	VARCHAR(128)	Имя параметра системы передачи сообщений.

Таблица sr_remotetable

Назначение Каждая строка описывает таблицу, отмеченную для репликации с использованием SQL Remote.

Столбцы	Столбец	Тип данных	Описание
	table_id	INT	Идентификатор таблицы.
	resolve_name	VARCHAR(128)	Имя хранимой процедуры, которая должна выполняться в случаях конфликтов.
	old_row_name	VARCHAR(128)	Таблица, которая содержит старое имя строки.
	remote_row_name	VARCHAR(128)	Таблица, которая содержит имя строки в удаленной базе данных.

Таблица sr_remotetype

Назначение Каждая строка описывает один из типов сообщений SQL Remote, включая адрес издателя.

Столбцы	Столбец	Тип данных	Описание
	type_id	INT	Идентификатор типа сообщений.
	type_name	VARCHAR(128)	Тип сообщений. Для каждого из указанных ниже параметров имеется отдельная строка: <ul style="list-style-type: none"> ◆ FILE ◆ MAPI ◆ VIM ◆ SMTP
	publisher_address	VARCHAR(128)	Адрес издателя для типа сообщений type_name .

Таблица sr_remotouser

Назначение

Каждая строка описывает идентификатор пользователя с полномочиями REMOTE (подписчика) вместе с состоянием сообщений SQL Remote, отправленных этому пользователю и полученных от него.

Столбцы

Столбец	Тип данных	Описание
user_id	INT	Идентификатор пользователя, имеющего полномочия REMOTE.
consolidate	CHAR(1)	Столбец содержит либо N, что указывает на наличие предоставленных пользователю полномочий REMOTE, либо Y, что указывает на наличие предоставленных пользователю полномочий CONSOLIDATE.
type_id	INT	Идентификатор системы передачи сообщений, которая использовалась для отправки сообщений этому пользователю.
address	VARCHAR(128)	Адрес, на который должны отправляться сообщения SQL Remote. Адрес должен соответствовать address_type .
frequency	CHAR(1)	Периодичность отправки сообщений SQL Remote.
send_time	DATETIME	Указание времени следующей отправки сообщений данному пользователю.
log_send	NUMERIC(20, 0)	Сообщения отправляются только таким подписчикам, для которых log_send больше, чем log_sent .
time_sent	DATETIME	Время отправки последнего сообщения данному подписчику.
log_sent	NUMERIC(20, 0)	Смещение в журнале, соответствующее последней операции отправки.
confirm_sent	NUMERIC(20, 0)	Смещение в журнале, соответствующее последней подтвержденной данным подписчиком операции.
send_count	INT	Количество сообщений, отправленных SQL Remote.
resend_count	INT	Счетчик, обеспечивающий однократную обработку сообщений в базе данных подписчика.
time_received	DATETIME	Время получения последнего сообщения от данного подписчика.
log_received	NUMERIC(20, 0)	Смещение в журнале базы данных подписчика, соответствующее операции, полученной в текущей базе данных последней.
confirm_received	NUMERIC(20, 0)	Смещение в журнале базы данных

Столбцы	Столбец	Тип данных	Описание
			подписчика, соответствующее операции, подтверждение которой было отправлено последним.
	receive_count	INT	Количество сообщений, полученных от данного подписчика.
	rereceive_count	INT	Счетчик, обеспечивающий однократную обработку сообщений в текущей базе данных.
	filler1	CHAR(255)	Зарезервирован
	filler2	CHAR(255)	Зарезервирован
	filler3	CHAR(255)	Зарезервирован
	filler4	CHAR(255)	Зарезервирован

Таблица sr_subscription

Назначение Каждая строка описывает подписку для одного идентификатора пользователя (пользователь должен иметь полномочия REMOTE) на одну публикацию.

Столбцы	Столбец	Тип данных	Описание
	publication_id	INT	Идентификатор публикации, на которую подписан пользователь.
	user_id	INT	Идентификатор пользователя, подписанного на публикацию.
	subscribe_by	VARCHAR(128)	Этот столбец содержит значение соответствия для данной подписки для публикаций с выражением SUBSCRIBE BY.
	created	NUMERIC(20, 0)	Смещение в журнале транзакций, при котором подписка была создана.
	started	NUMERIC(20, 0)	Смещение в журнале транзакций, при котором подписка была активизирована.
	operation	VARCHAR(20)	

Системные представления SQL Remote

Данный раздел описывает представления базы данных, используемые в SQL Remote для вывода и организации выводимой информации.

Представление sr_articles

Назначение

Каждая строка описывает статью.

Столбцы

Столбец	Описание
publication_name	Публикация, частью которой является данная статья.
table_name	Каждая статья состоит из столбцов и строк из одной таблицы. Данный столбец содержит имя этой таблицы.
where_expr	Этот столбец содержит условие поиска для статей, которые содержат поднабор строк, определенных разделом WHERE.
subscribe_by_expr	Этот столбец содержит выражение для статей, которые содержат поднабор строк, определенных выражением SUBSCRIBE BY.
subscribe_by_view	Этот столбец содержит имя представления для статей, которые содержат поднабор строк, определенных представлением.

Представление sr_articlecols

Назначение

Каждая строка описывает столбец, который появляется в статье.

Столбцы

Столбец	Описание
publication_name	Имя публикации, частью которой является столбец.
table_name	Имя таблицы, которой принадлежит столбец.
column_name	Имя столбца.

Представление sr_publications

Назначение

Перечисление имен всех публикаций.

Столбцы

Столбец	Описание
publication_name	Имя публикации.

Представление sr_remoteoptions

Назначение

Перечисление в более удобной для прочтения форме параметров системы передачи сообщений SQL Remote и их значений, хранимых в системных таблицах *remoteoption* и *remoteoptiontype*.

Столбцы	Столбец	Описание
	type_name	Тип системы передачи сообщений.
	"option"	Имя параметра.
	setting	Значение параметра.

Представление sr_remotetables

Назначение Перечисление в более удобной для прочтения форме таблиц, отмеченных для репликации SQL Remote, сохраненных в системной таблице **remotetable**.

Эта таблица существует только в Adaptive Server Enterprise.

Столбцы	Столбец	Описание
	table_name	Имя таблицы.
	resolve_name	Имя хранимой процедуры, которая должна выполняться в случаях конфликтов.
	old_row_name	Таблица, которая содержит старое имя строки.
	remote_row_name	Таблица, которая содержит имя строки в удаленной базе данных.

Представление sr_remotetypes

Назначение Перечисление типов сообщений, сохраненных в системной таблице **remotetype**.

Столбцы	Столбец	Описание
	type_id	Идентификатор типа сообщений.
	type_name	Тип сообщений. Для каждого из указанных ниже параметров имеется отдельная строка: <ul style="list-style-type: none"> ◆ FILE ◆ MAPI ◆ VIM ◆ SMTP
	publisher_address	Адрес издателя для типа сообщений type_name .

Представление sr_remoteusers

Назначение Вывод информации об удаленных пользователях и их статусе.

Столбцы	Столбец	Описание
	user_name	Идентификатор пользователя с полномочиями REMOTE.
	consolidate	Столбец содержит либо N, что указывает на наличие предоставленных пользователю полномочий REMOTE, либо Y, что указывает на наличие предоставленных пользователю полномочий CONSOLIDATE.
	type_name	Имя системы передачи сообщений, используемой для отправки сообщений данному пользователю.

Столбцы	Столбец	Описание
	address	Адрес, на который должны отправляться сообщения SQL Remote. Адрес должен соответствовать address_type .
	frequency	Периодичность отправки сообщений SQL Remote.
	send_time	Указание времени следующей отправки сообщений данному пользователю.
	next_send	Указание времени следующей отправки сообщений данному пользователю в более удобной для прочтения форме.
	log_send	Сообщения отправляются только тем подписчикам, для которых log_send больше, чем log_sent .
	time_sent	Время отправки последнего сообщения данному подписчику.
	log_sent	Смещение в журнале, соответствующее последней операции отправки.
	confirm_sent	Смещение в журнале, соответствующее последней подтвержденной данным подписчиком операции.
	send_count	Количество сообщений, отправленных SQL Remote.
	resend_count	Счетчик, обеспечивающий однократную обработку сообщений в базе данных подписчика.
	time_received	Время получения последнего сообщения от данного подписчика.
	log_received	Смещение в журнале базы данных подписчика, соответствующее операции, полученной в текущей базе данных последней.
	confirm_received	Смещение в журнале базы данных подписчика, соответствующее операции, подтверждение которой было отправлено последним.
	receive_count	Количество полученных сообщений.
	rereceive_count	Счетчик, обеспечивающий однократную обработку сообщений в текущей базе данных.

Представление sr_subscriptions

Назначение Каждая строка представляет информацию о подписке.

Столбцы	Столбец	Описание
	publication_name	Имя публикации, на которую подписан пользователь.
	user_name	Идентификатор пользователя, подписанного на публикацию.
	subscribe_by	Этот столбец содержит значение соответствия для данной подписки для публикаций с выражением SUBSCRIBE BY.

Столбцы	Столбец	Описание
	created	Смещение в журнале транзакций, при котором подписка была создана.
	started	Смещение в журнале транзакций, при котором подписка была активизирована.

Таблицы очереди с сохранением

В данном разделе описаны таблицы базы данных, используемые SQL Remote для определения и управления данными очереди с сохранением. Очередь с сохранением может быть находиться в той же базе данных, в которой находится база данных SQL Remote, либо в отдельной базе данных.

Очередь с сохранением используется только SQL Remote для Adaptive Server Enterprise.

Таблица sr_queue_state

Назначение Таблица из одной строки, в которой сохраняется постоянная глобальная информация о состоянии очереди с сохранением.

Столбцы	Столбец	Тип данных	Описание
	version	INT	Номер версии очереди с сохранением
	page_id	INT	page_id последней сосканированной записи в журнале транзакций.
	row_id	INT	row_id последней сосканированной записи в журнале транзакций.
	confirm_offset	NUMERIC(20,0)	Минимальное значение смещений подтверждений, полученных от всех удаленных пользователей. Это значение используется Message Agent для принятия решения о том, какие транзакции могут быть удалены из очереди с сохранением.
	commit_offset	NUMERIC(20,0)	Смещение в журнале транзакций для последней транзакции, завершенной раньше, чем самая старая незавершенная транзакция.
	backup_offset	NUMERIC(20,0)	Смещение в журнале транзакций для последней выполненной команды дампа базы данных или дампа транзакций . Данная информация используется в случае запуска Message Agent с параметром -u (репликация только транзакций с резервными копиями).
	marker	DATETIME	Последнее входящее сообщение, которое было сосканировано в очередь с сохранением. Задание столбца <code>time_received</code> таблицы <code>sr_remoteuser</code> осуществляется при обработке сообщения на сервере Adaptive Server Enterprise. Задание столбца <code>time_received</code> таблицы <code>sr_queue_state</code> осуществляется по окончании сканирования журнала транзакций и записи транзакций из этого сообщения в очередь с сохранением. Столбец предназначен для координации действий между одним Message Agent, непрерывно сканирующим журнал транзакций, и другим Message Agent, который получает сообщения и

confirmed_id	NUMERIC(20,0)	<p>отправляет сообщения в пакетном режиме. При использовании пакетного режима Message Agent получает сообщения, ожидает сканирования этих сообщений в очередь с сохранением, а затем осуществляет их отправку.</p> <p>Состояние ожидания устанавливается во всей базе данных по значению столбца time_received в таблице sr_queue_state.</p> <p>Поток отправки удаляет строки с меньшим confirmed_id, чем значение из sr_confirmed_transaction.</p>
---------------------	---------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Таблица sr_transaction

Назначение Данная таблица имеет одну строку для каждой транзакции в очереди с сохранением.

Столбец	Столбец	Описание
	offset	Смещение в журнале транзакций, соответствующее операции подтверждения транзакции. Это значение уникально идентифицирует каждую транзакцию.
	user_id	Удаленный пользователь узла, в котором была инициирована транзакция. Если транзакция не была инициирована удаленным пользователем, значение этого столбца устанавливается в NULL.
	data	Столбец user_id используется для предотвращения обратной репликации действий в удаленный узел, который их отправил. Непосредственно транзакция (внутренняя форма).

Таблица sr_confirmed_transaction

Назначение Каждая строка отмечает соответствующую строку в **sr_transaction**.

Столбец	Столбец	Тип данных	Описание
	confirmed_id	NUMERIC(20,0)	Уникальный идентификатор
	offset	NUMERIC(20,0)	Копия смещения, используемого для того, чтобы отмечать строки в sr_transaction для удаления.

Таблица sr_queue_coordinate

Назначение Одна строка, координирующая поток сканирования журнала SQL Remote и поток отправки для обращения к очереди с сохранением и связанным таблицам.

Столбец	Столбец	Тип данных	Описание
	status	CHAR(1)	Если очередь с сохранением еще не использовалась SQL Remote, устанавливается в значение N. Значения I и S устанавливаются при обращении потока сканирования журнала SQL Remote и потока отправки к очереди.

Справочник по командам Adaptive Server Anywhere

Об этой главе

В данной главе описаны операторы SQL, используемые для выполнения команд SQL Remote, и системные таблицы, используемые для хранения информации о системе репликации SQL Remote и ее состоянии.

Содержание

Раздел	Страница
Оператор ALTER REMOTE MESSAGE TYPE	304
Оператор CREATE PUBLICATION	305
Оператор CREATE REMOTE MESSAGE TYPE	306
Оператор CREATE SUBSCRIPTION	307
Оператор CREATE TRIGGER	308
Оператор DROP PUBLICATION	310
Оператор DROP REMOTE MESSAGE TYPE	311
Оператор DROP SUBSCRIPTION	312
Оператор GRANT CONSOLIDATE	313
Оператор GRANT PUBLISH	314
Оператор GRANT REMOTE	315
Оператор GRANT REMOTE DBA	316
Оператор PASSTHROUGH	317
Оператор REMOTE RESET	318
Оператор REVOKE CONSOLIDATE	319
Оператор REVOKE PUBLISH	320
Оператор REVOKE REMOTE	321
Оператор REVOKE REMOTE DBA	322
Оператор SET REMOTE OPTION	323
Оператор START SUBSCRIPTION	324
Оператор STOP SUBSCRIPTION	325
Оператор SYNCHRONIZE SUBSCRIPTION	326
Оператор UPDATE	327

Оператор ALTER REMOTE MESSAGE TYPE

Назначение Данный оператор используется для изменения системы передачи сообщений издателя, либо для изменения адреса издателя для указанной системы передачи сообщений, назначаемой для создаваемого типа сообщений.

Синтаксис **ALTER REMOTE MESSAGE TYPE** *система-передачи-сообщений*
ADDRESS *адрес*

система-передачи-сообщений: **FILE** | **FTP** | **MAPI** | **SMTP** | **VIM**

адрес: строка

Параметры	Параметр	Описание
	<i>система-передачи-сообщений</i>	Одна из систем передачи сообщений, поддерживаемых SQL Remote. Значением может быть одно из следующих:
	<i>адрес</i>	Строка, содержащая действительный адрес для указанной системы передачи сообщений.

Полномочия Требуются полномочия администратора БД.

Побочный эффект Автоматическое подтверждение.

Дополнительная информация "Оператор ALTER REMOTE MESSAGE TYPE [SQL Remote]" (ALTER REMOTE MESSAGE TYPE statement [SQL Remote]) на стр. 218 в документе "Справочник по SQL для ASA" (ASA SQL Reference Manual),
 "Оператор CREATE REMOTE MESSAGE TYPE" на стр. 306,
 "Процедура sp_remote_type" на стр. 373.

Оператор CREATE PUBLICATION

Назначение	Данный оператор используется для создания публикаций. Публикации в SQL Remote служат для определения реплицируемых данных в консолидированной и удаленной базах данных.
Синтаксис	CREATE PUBLICATION [<i>владелец</i> .] <i>имя-публикации</i> (TABLE <i>описание-статьи</i> , ...) <i>владелец</i> , <i>имя-публикации</i> : идентификатор <i>описание-статьи</i> : <i>имя-таблицы</i> [(<i>имя-столбца</i> , ...)] [WHERE <i>условие-поиска</i>] [SUBSCRIBE BY <i>выражение</i>]
Дополнительная информация	"Оператор CREATE PUBLICATION" (CREATE PUBLICATION statement) на стр. 314 в документе "Справочник по SQL для ASA" (ASA SQL Reference Manual).

Оператор CREATE REMOTE MESSAGE TYPE

Назначение Данный оператор используется для определения системы передачи сообщений и адреса возврата для исходящих сообщений от базы данных.

Синтаксис **CREATE REMOTE MESSAGE TYPE** *система-передачи-сообщений*
ADDRESS *адрес*

система-передачи-сообщений: **FILE** | **FTP** | **MAPI** | **SMTP** | **VIM**

адрес: строка

Параметры

Параметр	Описание
<i>система-передачи-сообщений</i>	Одна из поддерживаемых систем передачи сообщений.
<i>адрес</i>	Адрес для указанной системы передачи сообщений.

Полномочия Требуется полномочия администратора БД.

Побочные эффекты Автоматическое подтверждение.

Дополнительная информация "Оператор CREATE REMOTE MESSAGE TYPE [SQL Remote]" (CREATE REMOTE MESSAGE TYPE statement [SQL Remote]) на стр. 317 в документе "*Справочник по SQL для ASA*" (*ASA SQL Reference Manual*),
"Оператор GRANT PUBLISH" на стр. 314,
"Оператор GRANT REMOTE" на стр. 315,
"Оператор GRANT CONSOLIDATE" на стр. 313,
"Процедура sp_remote_type" на стр. 373,
"Управление типами сообщений" на стр. 183.

Оператор CREATE SUBSCRIPTION

Назначение Данный оператор используется для создания подписки пользователя на публикацию.

Синтаксис **CREATE SUBSCRIPTION**
TO *имя-публикации* [(*значение-подписки*)]
FOR *идентификатор-подписчика*
имя-публикации: идентификатор
значение-подписки, идентификатор-подписчика: строка
идентификатор-подписчика: строка

Параметры	Параметр	Описание
	<i>имя-подписки</i>	Имя публикации, на которую подписывается пользователь. Может включать владельца публикации.
	<i>значение-подписки</i>	Строка, которая сравнивается с выражением подписки на публикацию. Подписчик получает все строки, для которых выражение подписки соответствует значению подписки.
	<i>идентификатор-подписчика</i>	Идентификатор подписчика на публикацию. Пользователю должны быть предоставлены полномочия REMOTE.

Полномочия Требуется полномочия администратора БД.

Побочные эффекты Автоматическое подтверждение.

Дополнительная информация "Оператор CREATE SUBSCRIPTION [SQL Remote]" (CREATE SUBSCRIPTION statement [SQL Remote]) на стр. 324 в документе "*Справочник по SQL для ASA*" (*ASA SQL Reference Manual*).
"Оператор DROP SUBSCRIPTION" на стр. 312,
"Оператор GRANT REMOTE" на стр. 315,
"Оператор SYNCHRONIZE SUBSCRIPTION" на стр. 326,
"Оператор START SUBSCRIPTION" на стр. 324,
"Процедура sp_subscription" на стр. 379.

Оператор CREATE TRIGGER

Назначение	Данный оператор используется для создания в базе данных нового триггера. Одна из форм триггера специально предназначена для использования SQL Remote.
Синтаксис	<pre> CREATE TRIGGER <i>имя-триггера</i> <i>время-триггера</i> <i>событие-триггера</i>, ... [ORDER <i>integer</i>] ON <i>имя-таблицы</i> [REFERENCING [OLD AS <i>старое-имя</i>] [NEW AS <i>новое-имя</i>]] [REMOTE AS <i>имя-удаленной-БД</i>]] [FOR EACH { ROW STATEMENT }] [WHEN (<i>условие-поиска</i>)] [IF UPDATE (<i>имя-столбца</i>) THEN [{ AND OR } UPDATE (<i>имя-столбца</i>)] ...] <i>составной-оператор</i> [ELSEIF UPDATE (<i>имя-столбца</i>) THEN [{ AND OR } UPDATE (<i>имя-столбца</i>)] ...] <i>составной-оператор</i> END IF]] </pre> <p><i>время-триггера:</i> BEFORE AFTER RESOLVE</p> <p><i>событие-триггера:</i> DELETE INSERT UPDATE UPDATE OF <i>имя-столбца</i> [, <i>имя-столбца</i>, ...]</p>
Параметры	<p>время-триггера Триггеры уровня строк могут быть определены для выполнения до (BEFORE) или после (AFTER) вставки, обновления или удаления. Триггеры уровня операторов выполняются после (AFTER) операторов. Время триггера RESOLVE используется в SQL Remote для запуска триггера для данного списка столбцов только перед операциями UPDATE или UPDATE OF уровня строк.</p> <p>Триггеры BEFORE UPDATE запускаются каждый раз при выполнении операции UPDATE в отношении строки независимо от того, отличается ли новое значение от старого. Триггеры AFTER UPDATE запускаются только тогда, когда новые значения отличаются от старых значений.</p> <p>События триггера. Триггеры могут быть запущены при одном или нескольких событиях:</p> <ul style="list-style-type: none"> ◆ DELETE Вызывается всякий раз при удалении строки связанной таблицы ◆ INSERT Вызывается всякий раз при вставке новой строки в таблицу, связанную с триггером. ◆ UPDATE Вызывается всякий раз при обновлении строки связанной таблицы ◆ UPDATE OF список-столбцов Вызывается всякий раз, когда обновляется строка связанной таблицы и модифицируется столбец из указанного <i>списка-столбцов</i>.
Использование	Без ограничений.
Полномочия	Требуются полномочия RESOURCE и полномочия ALTER на таблицу, либо полномочия администратора БД. CREATE TRIGGER блокировку данной таблицы и таким образом устанавливает режим монопольного использования таблицы.
Побочные эффекты	Автоматическое подтверждение.
Дополнительная информация	"Оператор CREATE TRIGGER [SQL Remote]" (CREATE TRIGGER statement [SQL Remote]) на стр. 366 в документе "Справочник по SQL для ASA" (ASA SQL Reference Manual).

"Оператор UPDATE" на стр. 327.

Оператор DROP PUBLICATION

Назначение	Данный оператор используется для удаления публикаций. Публикации в SQL Remote служат для определения реплицируемых данных в консолидированной и удаленной базах данных.
Синтаксис	DROP PUBLICATION [<i>владелец</i> .] <i>имя-публикации</i> <i>владелец, имя-публикации</i> : идентификатор
Дополнительная информация	"Оператор DROP PUBLICATION" (DROP PUBLICATION statement) на стр. 402 в документе " <i>Справочник по SQL для ASA</i> " (<i>ASA SQL Reference Manual</i>)

Оператор DROP REMOTE MESSAGE TYPE

Назначение	Данный оператор используется для удаления определений типа сообщений из базы данных.					
Синтаксис	DROP REMOTE MESSAGE TYPE <i>система-передачи-сообщений</i> <i>система-передачи-сообщений</i> : FILE FTP MAPI SMTP VIM					
Параметры	<table border="1"> <thead> <tr> <th>Параметр</th> <th>Описание</th> </tr> </thead> <tbody> <tr> <td><i>система-передачи-сообщений</i></td> <td>Одна из систем передачи сообщений, поддерживаемых SQL Remote.</td> </tr> </tbody> </table>	Параметр	Описание	<i>система-передачи-сообщений</i>	Одна из систем передачи сообщений, поддерживаемых SQL Remote.	
Параметр	Описание					
<i>система-передачи-сообщений</i>	Одна из систем передачи сообщений, поддерживаемых SQL Remote.					
Полномочия	Требуются полномочия администратора БД. Удаление типа сообщений возможно только при отсутствии пользователей этого типа сообщений с полномочиями REMOTE или CONSOLIDATE.					
Побочные эффекты	Автоматическое подтверждение.					
Дополнительная информация	<p>"Оператор DROP REMOTE MESSAGE TYPE" (DROP REMOTE MESSAGE TYPE statement) на стр. 403 в документе <i>"Справочник по SQL для ASA" (ASA SQL Reference Manual)</i>,</p> <p>"Оператор CREATE REMOTE MESSAGE TYPE" на стр. 306,</p> <p>"Оператор ALTER REMOTE MESSAGE TYPE" на стр. 361,</p> <p>"Процедура sp_drop_remote_type" на стр. 338,</p> <p>"Управление типами сообщений" на стр. 183.</p>					

Оператор DROP SUBSCRIPTION

Назначение Данный оператор используется для удаления подписки пользователя на публикацию.

Синтаксис **DROP SUBSCRIPTION TO** *имя-публикации* [(*значение-подписки*)]
FOR *идентификатор-подписчика*, ...

значение-подписки: строка

имя-подписчика: строка

Параметры

Параметр	Описание
<i>имя-публикации</i>	Имя публикации, на которую подписывается пользователь. Может включать владельца публикации.
<i>значение-подписки</i>	Строка, которая сравнивается с выражением подписки публикации. Значение подписки требуется по причине того, что пользователь может иметь более одной подписку на публикации.
<i>идентификатор-подписчика</i>	Идентификатор подписчика на публикацию.

Полномочия Требуется полномочия администратора БД.

Побочные эффекты Автоматическое подтверждение.

Дополнительная информация "Оператор DROP SUBSCRIPTION" (DROP SUBSCRIPTION statement) на стр. 407 в документе *"Справочник по SQL для ASA" (ASA SQL Reference Manual)*,
 "Оператор CREATE SUBSCRIPTION" на стр. 307,
 "Оператор DROP SUBSCRIPTION" на стр. 312.

Оператор GRANT CONSOLIDATE

Назначение Данный оператор используется для определения базы данных, которая в иерархии SQL Remote находится непосредственно на один уровень выше текущей базы данных и которая будет получать сообщения от текущей базы данных.

Синтаксис

```
GRANT CONSOLIDATE
  TO идентификатор-пользователя, ...
  TYPE система-передачи-сообщений, ...
  ADDRESS строка-адреса, ...
  [SEND { EVERY | AT }'hh:mm' ]
```

система-передачи-сообщений: **FILE** | **FTP** | **MAPI** | **SMTP** | **VIM**

адрес: *строка*

Параметры	Параметр	Описание
	<i>идентификатор-пользователя</i>	Идентификатор пользователя, которому будут предоставлены полномочия
	<i>система-передачи-сообщений</i>	Одна из систем передачи сообщений, поддерживаемых SQL Remote.
	<i>адрес</i>	Адрес для указанной системы передачи сообщений.

Полномочия Требуется полномочия администратора БД.

Побочные эффекты Автоматическое подтверждение.

Дополнительная информация "Оператор GRANT CONSOLIDATE [SQL Remote]" (GRANT CONSOLIDATE statement [SQL Remote]) на стр. 447 в документе "*Справочник по SQL для ASA*" (*ASA SQL Reference Manual*),
 "Оператор GRANT REMOTE" на стр. 315,
 "Оператор REVOKE CONSOLIDATE" на стр. 319,
 "Оператор GRANT PUBLISH" на стр. 314,
 "Процедура sp_grant_consolidate" на стр. 340.

Оператор GRANT PUBLISH

Назначение	Данный оператор используется для определения издателя текущей базы данных.
Синтаксис	GRANT PUBLISH TO <i>идентификатор-пользователя</i>
Полномочия	Требуются полномочия администратора БД.
Побочные эффекты	Автоматическое подтверждение.
Дополнительная информация	"Оператор GRANT PUBLISH [SQL Remote]" (GRANT PUBLISH statement [SQL Remote]) на стр. 449 в документе "Справочник по SQL для ASA" (ASA SQL Reference Manual), "Оператор GRANT REMOTE" на стр. 315, "Оператор GRANT CONSOLIDATE" на стр. 313, "Оператор REVOKE PUBLISH" на стр. 320, "Оператор CREATE PUBLICATION" (CREATE PUBLICATION statement) на стр. 314 в документе "Справочник по SQL для ASA" (ASA SQL Reference Manual), "Оператор CREATE SUBSCRIPTION" на стр. 307, "Процедура sp_publisher" на стр. 356.

Оператор GRANT REMOTE

Назначение Данный оператор используется для определения базы данных, которая находится в иерархии SQL Remote непосредственно на один уровень ниже текущей базы данных и которая будет получать сообщения от текущей базы данных. Такие базы данных могут упоминаться как удаленные пользователи.

Синтаксис **GRANT REMOTE TO** *идентификатор-пользователя*, ...
TYPE *система-передачи-сообщений*, ...
ADDRESS *строка-адреса*, ...
 [**SEND** { **EVERY** | **AT** } *время-отправки*]

Параметры	Параметр	Описание
	<i>идентификатор-пользователя</i>	Идентификатор пользователя, которому будут предоставлены полномочия
	<i>система-передачи-сообщений</i>	Одна из систем передачи сообщений, поддерживаемых SQL Remote. Значением может быть одно из следующих: <ul style="list-style-type: none"> ◆ FILE ◆ FTP ◆ MAPI ◆ SMTP ◆ VIM
	<i>строка адреса</i>	Строка, содержащая действительный адрес для указанной системы передачи сообщений.
	<i>время отправки</i>	Строка, содержащая значение времени в форме <i>hh:mm</i> .

Полномочия Требуется полномочия администратора БД.

Побочные эффекты Автоматическое подтверждение.

Дополнительная информация "Оператор GRANT REMOTE [SQL Remote]" на стр. 450 в документе "Справочник по SQL для ASA" (ASA SQL Reference Manual),
 "Оператор GRANT CONSOLIDATE" на стр. 313,
 "Оператор REVOKE REMOTE" на стр. 321,
 "Оператор GRANT PUBLISH" на стр. 314,
 "Процедура sp_grant_remote" на стр. 342,
 "Предоставление и отмена полномочий REMOTE и CONSOLIDATE" на стр. 177.

Оператор GRANT REMOTE DBA

Назначение	Данный оператор используется для предоставления полномочий администратора БД указанному пользователю только при выполнении подключения из Message Agent.
Синтаксис	GRANT REMOTE DBA TO <i>идентификатор-пользователя</i> , ... IDENTIFIED BY <i>пароль</i>
Полномочия	Требуются полномочия администратора БД.
Побочные эффекты	Автоматическое подтверждение.
Дополнительная информация	"Оператор GRANT REMOTE DBA" (GRANT REMOTE DBA statement) на стр. 452 в документе " <i>Справочник по SQL для ASA</i> " (<i>ASA SQL Reference Manual</i>), "Message Agent и безопасность репликации" на стр. 211, "Оператор REVOKE REMOTE DBA" на стр. 322.

Оператор PASSTHROUGH

Назначение	Данный оператор используется для включения или выключения режима ретрансляции при администрировании SQL Remote. При использовании синтаксиса 1 и 2 происходит включение режима ретрансляции, в то время синтаксис 3 служит для прекращения ретрансляции.
Синтаксис 1	PASSTHROUGH [ONLY] FOR <i>идентификатор-пользователя</i> , ...
Синтаксис 2	PASSTHROUGH [ONLY] FOR SUBSCRIPTION TO [(<i>владелец</i>)]. <i>имя-публикации</i> [(<i>константа</i>)]
Синтаксис 3	PASSTHROUGH STOP
Полномочия	Требуются полномочия администратора БД.
Побочные эффекты	Отсутствуют.
Дополнительная информация	"ОПЕРАТОР PASSTHROUGH [SQL Remote]" (PASSTHROUGH statement [SQL Remote]) на стр. 494 в документе "Справочник по SQL для ASA" (ASA SQL Reference Manual), "Процедура sp_passthrough" на стр. 349.

Оператор REMOTE RESET

Назначение	Данный оператор используется в специальных процедурах извлечения баз данных с целью активизации всех подписок удаленного пользователя в пределах одной транзакции.
Синтаксис	REMOTE RESET <i>идентификатор-пользователя</i>
Полномочия	Требуются полномочия администратора БД.
Побочные эффекты	В результате выполнения данного оператора автоматического подтверждения не происходит.
Дополнительная информация	"Оператор REMOTE RESET [SQL Remote]" (REMOTE RESET statement [SQL Remote]) на стр. 506 в документе "Справочник по SQL для ASA" (ASA SQL Reference Manual), "Оператор START SUBSCRIPTION" на стр. 324.

Оператор REVOKE CONSOLIDATE

Назначение	Данный оператор используется для запрета получения консолидированной базой данных сообщений SQL Remote от этой базы данных.
Синтаксис	REVOKE CONSOLIDATE FROM <i>идентификатор-пользователя</i> , ...
Полномочия	Требуются полномочия администратора БД.
Побочные эффекты	Автоматическое подтверждение. Удаление всех подписок данного пользователя.
Дополнительная информация	"Оператор REVOKE CONSOLIDATE" (REVOKE CONSOLIDATE statement) на стр. 518 в документе " <i>Справочник по SQL для ASA</i> " (<i>ASA SQL Reference Manual</i>), "Оператор GRANT CONSOLIDATE" на стр. 313, "Процедура sp_revoke_consolidate" на стр. 377.

Оператор REVOKE PUBLISH

Назначение	Данный оператор используется для отмены определения данного пользователя как текущего (CURRENT) издателя.
Синтаксис	REVOKE PUBLISH FROM <i>идентификатор-пользователя</i>
Полномочия	Требуются полномочия администратора БД.
Побочные эффекты	Автоматическое подтверждение.
Дополнительная информация	"Оператор REVOKE PUBLISH [SQL Remote]" (REVOKE PUBLISH statement [SQL Remote]) на стр. 519 в документе "Справочник по SQL для ASA" (ASA SQL Reference Manual), "Оператор GRANT PUBLISH" на стр. 314, "Оператор REVOKE REMOTE" на стр. 321, "Оператор CREATE PUBLICATION" (CREATE PUBLICATION statement) на стр. 314 в документе "Справочник по SQL для ASA" (ASA SQL Reference Manual) "Справочник по SQL для ASA" (ASA SQL Reference Manual), "Оператор CREATE SUBSCRIPTION" на стр. 307, "Процедура sp_publisher" на стр. 356.

Оператор REVOKE REMOTE

Назначение	Данный оператор используется для запрета получения пользователем сообщений SQL Remote от этой базы данных.
Синтаксис	REVOKE REMOTE FROM <i>идентификатор-пользователя</i> , ...
Полномочия	Требуются полномочия администратора БД.
Побочные эффекты	Автоматическое подтверждение. Удаление всех подписок данного пользователя.
Дополнительная информация	"Оператор REVOKE REMOTE [SQL Remote]" (REVOKE REMOTE statement [SQL Remote]) на стр. 520 в документе "Справочник по SQL для ASA" (ASA SQL Reference Manual), "Процедура sp_revoke_remote" на стр. 378.

Оператор REVOKE REMOTE DBA

Назначение	Данный оператор используется для отмены полномочий администратора БД, назначенных указанному пользователю для выполнения подключения из Message Agent.
Синтаксис 1	REVOKE REMOTE DBA FROM <i>идентификатор-пользователя</i> , ...
Полномочия	Требуются полномочия администратора БД.
Побочные эффекты	Автоматическое подтверждение.
Дополнительная информация	"Оператор REVOKE REMOTE DBA" (REVOKE REMOTE DBA statement) на стр. 521 в документе " <i>Справочник по SQL для ASA</i> " (<i>ASA SQL Reference Manual</i>), "Message Agent и безопасность репликации" на стр. 211, "Оператор GRANT REMOTE DBA" на стр. 316.

Оператор SET REMOTE OPTION

Назначение	Данный оператор используется для установки параметров управления сообщениями для системы передачи сообщений SQL Remote.
Синтаксис	SET REMOTE <i>имя-системы</i> OPTION [<i>идентификатор-пользователя</i> . PUBLIC .] <i>имя-параметра-системы</i> = <i>значение-параметра-системы</i>
Параметры	<i>имя-системы</i> : file ftp mapi smtp vim <i>имя-параметра-системы</i> : <i>параметр-file</i> <i>параметр-ftp</i> <i>параметр-mapi</i> <i>параметр-smtp</i> <i>параметр-vim</i> <i>параметр-file</i> : debug directory <i>параметр-ftp</i> : active_mode debug host password port root_directory user <i>параметр-mapi</i> : debug force_download ipm_receive ipm_send profile <i>параметр-smtp</i> : debug local_host pop3_host pop3_password pop3_userid smtp_host top_supported <i>параметр-vim</i> : debug password path receive_all send_vim_mail userid <i>значение-параметра-системы</i> : <i>строка</i>
Полномочия	Требуются полномочия администратора БД. Издатель может задать собственные параметры.
Побочные эффекты	Автоматическое подтверждение.
Дополнительная информация	"Процедура sp_link_option" на стр. 344.

Оператор START SUBSCRIPTION

Назначение Данный оператор используется для активизации подписки пользователя на публикацию.

Синтаксис **START SUBSCRIPTION**
TO *имя-публикации* [(*значение-подписки*)]
FOR *идентификатор-подписчика*, ...

Параметры	Параметр	Описание
	<i>имя-публикации</i>	Имя публикации, на которую подписывается пользователь. Может включать владельца публикации.
	<i>значение-подписки</i>	Строка, которая сравнивается с выражением подписки на публикацию. Значение подписки требуется здесь по причине того, что каждый подписчик может иметь более одной подписки на публикацию.
	<i>идентификатор-подписчика</i>	Идентификатор подписчика на публикацию. Этот пользователь должен иметь подписку на публикацию.

Полномочия Требуется полномочия администратора БД.

Побочные эффекты Автоматическое подтверждение.

Дополнительная информация "Оператор START SUBSCRIPTION" (START SUBSCRIPTION statement) на стр. 554 в документе "*Справочник по SQL для ASA*" (*ASA SQL Reference Manual*), "Оператор CREATE SUBSCRIPTION" на стр. 307, "Оператор REMOTE RESET" на стр. 318, "Оператор SYNCHRONIZE SUBSCRIPTION" на стр. 326, "Процедура sp_subscription" на стр. 379.

Оператор STOP SUBSCRIPTION

Назначение Данный оператор используется для деактивизации подписки пользователя на публикацию.

Синтаксис **STOP SUBSCRIPTION**
TO *имя-публикации* [(*значение-подписки*)]
FOR *идентификатор-подписчика*, ...

Параметры	Параметр	Описание
	<i>имя-публикации</i>	Имя публикации, на которую подписывается пользователь. Может включать владельца публикации.
	<i>значение-подписки</i>	Строка, которая сравнивается с выражением подписки публикации. Значение подписки требуется здесь по причине того, что каждый подписчик может иметь более одной подписки на публикацию.
	<i>идентификатор-подписчика</i>	Идентификатор подписчика на публикацию. Данный пользователь должен иметь подписку на публикацию.

Полномочия Требуется полномочия администратора БД.

Побочные эффекты Автоматическое подтверждение.

Дополнительная информация "Оператор STOP SUBSCRIPTION [SQL Remote]" (STOP SUBSCRIPTION statement [SQL Remote]) на стр. 562 в документе "Справочник по SQL для ASA" (ASA SQL Reference Manual),
 "Оператор CREATE SUBSCRIPTION" на стр. 307,
 "Оператор SYNCHRONIZE SUBSCRIPTION" на стр. 326.

Оператор SYNCHRONIZE SUBSCRIPTION

Назначение Данный оператор используется для синхронизации подписки пользователя на публикацию.

Синтаксис **SYNCHRONIZE SUBSCRIPTION**
TO имя-публикации [(значение-подписки)]
FOR удаленный-пользователь, ...

Параметры	Параметр	Описание
	<i>имя-публикации</i>	Имя публикации, на которую подписывается пользователь. Может включать владельца публикации.
	<i>значение-подписки</i>	Строка, которая сравнивается с выражением подписки публикации. Значение подписки требуется здесь по причине того, что каждый подписчик может иметь более одной подписки на публикацию.
	<i>удаленный-пользователь</i>	Идентификатор подписчика на публикацию. Данный пользователь должен иметь подписку на публикацию.

Полномочия Требуется полномочия администратора БД.

Побочные эффекты Автоматическое подтверждение.

Дополнительная информация "Оператор SYNCHRONIZE SUBSCRIPTION" (SYNCHRONIZE SUBSCRIPTION statement) на стр. 564 в документе "*Справочник по SQL для ASA*" (*ASA SQL Reference Manual*),
 "Оператор CREATE SUBSCRIPTION" на стр. 307,
 "Оператор START SUBSCRIPTION" на стр. 324.

Оператор UPDATE

Назначение	Данный оператор используется для внесения изменений в данные в базе данных.
Синтаксис 1	UPDATE <i>список-таблиц</i> SET <i>имя-столбца</i> = <i>выражение</i> , ... [VERIFY (<i>имя-столбца</i> , ...) VALUES (<i>выражение</i> , ...)] [WHERE <i>условие-поиска</i>] [ORDER BY <i>выражение</i> [ASC DESC], ...]
Синтаксис 2	UPDATE <i>таблица</i> PUBLICATION <i>публикация</i> { SUBSCRIBE BY <i>выражение</i> OLD SUBSCRIBE BY <i>выражение</i> NEW SUBSCRIBE BY <i>выражение</i> } WHERE <i>условие-поиска</i> <i>выражение</i> : <i>значение</i> <i>подзапрос</i>
Использование	Синтаксис 1 и синтаксис 2 применимы только в SQL Remote. Синтаксис 2 без выражений OLD и NEW SUBSCRIBE BY должен использоваться в триггере BEFORE. Синтаксис 2 с выражениями OLD и NEW SUBSCRIBE BY может использоваться везде.
Полномочия	Требуются полномочия UPDATE на изменяемые столбцы.
Побочные эффекты	Отсутствуют.
Дополнительная информация	"Оператор UPDATE [SQL Remote]" (UPDATE statement [SQL Remote]) на стр. 582 в документе "Справочник по SQL для ASA" (ASA SQL Reference Manual), "Оператор CREATE TRIGGER" на стр. 308.

Справочник по командам для Adaptive Server Enterprise

Об этой главе

В данной главе описаны хранимые процедуры SQL Remote, используемые для выполнения команд SQL Remote.

Содержание

Раздел	Страница
Процедура sp_add_article	331
Процедура sp_add_article_col	333
Процедура sp_add_remote_table	334
Процедура sp_create_publication	336
Процедура sp_drop_publication	337
Процедура sp_drop_remote_type	338
Процедура sp_drop_sql_remote	339
Процедура sp_grant Consolidate	340
Процедура sp_grant_remote	342
Процедура sp_link_option	344
Процедура sp_modify_article	346
Процедура sp_modify_remote_table	348
Процедура sp_passthrough	349
Процедура sp_passthrough_piece	350
Процедура sp_passthrough_stop	352
Процедура sp_passthrough_subscription	353
Процедура sp_passthrough_user	354
Процедура sp_populate_sql_anywhere	355
Процедура sp_publisher	356
Процедура sp_queue_clean	357
Процедура sp_queue_confirmed_delete_old	358
Процедура sp_queue_confirmed_transaction	359
Процедура sp_queue_delete_old	360
Процедура sp_queue_drop	361
Процедура sp_queue_dump_database	362
Процедура sp_queue_dump_transaction	363
Процедура sp_queue_get_state	364
Процедура sp_queue_log_transfer_reset	365
Процедура sp_queue_read	366

Процедура sp_queue_reset	367
Процедура sp_queue_set_confirm	368
Процедура sp_queue_set_progress	369
Процедура sp_queue_transaction	370
Процедура sp_remote	371
Процедура sp_remote_option	372
Процедура sp_remote_type	373
Процедура sp_remove_article	374
Процедура sp_remove_article_col	375
Процедура sp_remove_remote_table	376
Процедура sp_revoke_consolidate	377
Процедура sp_revoke_remote	378
Процедура sp_subscription	379
Процедура sp_subscription_reset	380

Процедура `sp_add_article`

Назначение Добавление статьи к публикации.

Синтаксис `sp_add_article имя_публикации, имя_таблицы, выражение_where, выражение_subscribe_by, представление_subscribe_by`

Аргумент	Описание
<code>имя_публикации</code>	Имя публикации, к которой должна быть добавлена статья.
<code>имя_таблицы</code>	Таблица, содержащая статью.
<code>выражение_where</code>	Этот необязательный аргумент должен быть именем столбца или иметь значение NULL. Публикация включает только те строки, для которых указанное значение столбца - не NULL. Значение по умолчанию - NULL, что означает, что никакие строки из публикации не исключаются.
<code>выражение_subscribe_by</code>	Новое выражение подписки, определяющее, какие строки должны быть включены в публикацию для каждой подписки. Выражение должно быть именем столбца в таблице, указанной как <code>имя_таблицы</code> . Значение по умолчанию - NULL.
<code>представление_subscribe_by</code>	Представление, определяющее столбцы и строки, которые будут включены в публикацию. ☞ Для получения дополнительной информации см. разделы "Повышение производительности при операции извлечения" на стр. 134 и "Повышение производительности при операции извлечения совместно используемых строк" на стр. 140.

Дополнительная информация "Процедура `sp_add_remote_table`" на стр. 334,
"Процедура `sp_create_publication`" на стр. 336,
"Процедура `sp_remove_article`" на стр. 374,
"Оператор CREATE PUBLICATION" (CREATE PUBLICATION statement) на стр. 314 в документе "Справочник по SQL для ASA" (ASA SQL Reference Manual).

Описание Процедура `sp_add_article` добавляет статью к публикации. Прежде чем таблица может быть добавлена к публикации, она быть отмечена для репликации с помощью процедуры `sp_add_remote_table`; невыполнение данного требования приводит к ошибке.

При вызове процедуры `sp_add_article` в публикацию добавляются все столбцы таблицы. При необходимости включения в публикацию лишь некоторых столбцов таблицы после выполнения `sp_add_article` необходимо вызвать `sp_add_article_col`.

Как и другие изменения в определениях данных, в реальной ситуации данная процедура должна выполняться только при минимальной нагрузке системы SQL Remote.

☞ Для получения дополнительной информации о требованиях по минимальной нагрузке см. раздел "Внесение изменений в схему" на стр. 284.

Пример ♦ Нижеприведенный оператор добавляет таблицу SalesRep к публикации с именем SalesRepData:

Процедура sp_add_article

```
sp_add_article 'SalesRepData', 'SalesRep'  
go
```

Процедура `sp_add_article_col`

Назначение Добавление столбца к статье в публикации.

Синтаксис `sp_add_article_col имя_публикации, имя_таблицы, имя_столбца`

Аргумент	Описание
<code>имя_публикации</code>	Имя публикации, к которой должна быть добавлена статья.
<code>имя_таблицы</code>	Таблица, содержащая статью.
<code>имя_столбца</code>	Столбец, который будет добавлен к статье в публикации.

Дополнительная информация "Процедура `sp_add_article`" на стр. 331,
 "Процедура `sp_remove_article`" на стр. 374,
 "Оператор ALTER PUBLICATION" (ALTER PUBLICATION statement) на стр. 216
 в документе *"Справочник по SQL для ASA"* (ASA SQL Reference Manual).

Описание Процедура `sp_add_article_col` добавляет столбец к статье в публикации. Для этого сначала следует добавить таблицу к публикации с помощью процедуры `sp_add_article` (см. раздел "Процедура `sp_add_article`" на стр. 331).

Для добавления к публикации всех столбцов таблицы нет необходимости использовать `sp_add_article_col`, достаточно вызова процедуры `sp_add_article`.

Чтобы добавить к публикации только некоторые столбцы таблицы, сначала вызывается `sp_add_article`, а затем - `sp_add_article_col` для каждого из столбцов, которые нужно включить в публикацию.

Как и другие изменения в определениях данных, в реальной ситуации данная процедура должна выполняться только при минимальной нагрузке системы SQL Remote.

☞ Для получения дополнительной информации о требованиях по минимальной нагрузке см. раздел "Внесение изменений в схему" на стр. 284.

Пример ♦ Нижеприведенные операторы добавляют столбцы `emp_id` и `emp_lname` таблицы `employee` к публикации с именем `Personnel`:

```
sp_add_article 'Personnel', employee'
sp_add_article_col 'Personnel', 'employee', 'emp_id'
sp_add_article_col 'Personnel', 'employee',
'emp_lname'
go
```

Процедура `sp_add_remote_table`

Назначение Процедура позволяет отметить таблицу для репликации SQL Remote.

Синтаксис `sp_add_remote_table` *имя_таблицы*,
 [*процедура_разрешения*,]
 [*старое_имя_строки*],
 [*имя_строки_удаленной_БД*]

Аргумент	Описание
<i>имя_таблицы</i>	Таблица, которая будет отмечена для репликации SQL Remote.
<i>процедура_разрешения</i>	Имя хранимой процедуры, выполняющей действия по разрешению возникнувших конфликтов.
<i>старое_имя_строки</i>	Имя таблицы, в которой хранятся значения при возникновении конфликта.
<i>имя_строки_удаленной_БД</i>	Имя таблицы, в которой хранятся значения удаленной базы данных при обработке вызвавшего конфликт оператора UPDATE.

Полномочия Эта процедура может выполняться только системным администратором.

Дополнительная информация "Процедура `sp_modify_remote_table`" на стр. 348,
 "Процедура `sp_remove_remote_table`" на стр. 376,
 "Управление конфликтами" на стр. 143.

Описание Прежде чем база данных может быть включена в любые публикации SQL Remote, каждая ее таблица должна быть отмечена для репликации с помощью `sp_add_remote_table`. После выполнения `sp_add_remote_table` таблица может быть добавлена к публикации при использовании процедур `sp_add_article` (см. раздел "Процедура `sp_add_article`" на стр. 331) и `sp_add_article_col` (см. раздел "Процедура `sp_add_article_col`" на стр. 333).

Процедура `sp_add_remote_table` вызывает процедуру `sp_setreplicate`, которая отмечает таблицу для репликации. На основании этого Adaptive Server Enterprise помещает в журнал транзакций расширенную информацию. Расширенная информация включает в себя полные образы строки до и после обновления.

Первый аргумент - имя таблицы, которая будет отмечена для репликации.

Остальные три аргумента являются необязательными. Они являются именами объектов, требуемых только для пользовательских методов разрешения конфликтов. Для пользовательских методов разрешения конфликтов необходимо указать имена двух таблиц и хранимой процедуры. Процедура `sp_add_remote_table` не проверяет существование объектов разрешения конфликтов: их можно создать как до, так и после отмечания таблицы для репликации.

Эти две таблицы должны иметь те же самые столбцы и типы данных, что и таблица, указанная как *имя_таблицы*.

Примеры

- ◆ Нижеприведенный оператор отмечает для репликации таблицу Customer, используя значения для разрешения конфликтов по умолчанию:

```
exec sp_add_remote_table Customer
```

- ◆ Нижеприведенный оператор отмечает для репликации таблицу Customer, используя для разрешения конфликтов хранимую процедуру Customer_Conflict. Старые и удаленные строки сохраняются в таблицах с именами old_Customer и remote_Customer соответственно:

```
exec sp_add_remote_table Customer,
```

Customer_Conflict, old_Customer, remote_Customer

Процедура **sp_create_publication**

Назначение Создание публикации.

Синтаксис **sp_create_publication** *имя_публикации*

Аргумент	Описание
<i>имя_публикации</i>	Имя создаваемой публикации.

Дополнительная информация "Процедура *sp_drop_publication*" на стр. 337,
"Оператор CREATE PUBLICATION" (CREATE PUBLICATION statement) на стр. 314 в документе *"Справочник по SQL для ASA"* (ASA SQL Reference Manual).

Описание Процедура **sp_create_publication** создает пустую публикацию (без статей). В созданную публикацию необходимо добавить статьи с помощью процедур *sp_add_remote_table* (см. раздел "Процедура *sp_add_remote_table*" на стр. 334) и *sp_add_article* (см. раздел "Процедура *sp_add_article*" на стр. 331).

Пример ♦ Нижеприведенный оператор создает публикацию с именем **SalesRepData**:

```
sp_create_publication 'SalesRepData'  
go
```


Процедура `sp_drop_publication`

Назначение Удаление публикации из базы данных.

Синтаксис `sp_drop_publication имя_публикации`

Аргумент	Описание
<code>имя_публикации</code>	Имя публикации, которая будет удалена.

Дополнительная информация "Процедура `sp_create_publication`" на стр. 336, "Оператор DROP PUBLICATION" (DROP PUBLICATION statement) на стр. 402 в документе *"Справочник по SQL для ASA"* (ASA SQL Reference Manual).

Описание Процедура `sp_drop_publication` удаляет публикацию из базы данных. Также удаляются все статьи, составляющие данную публикацию, и подписки на публикацию.

Пример ♦ Нижеприведенный оператор удаляет публикацию SalesRep:

```
sp_drop_publication 'SalesRep'
go
```

Процедура `sp_drop_remote_type`

Назначение Удаление типа сообщений из базы данных.

Синтаксис `sp_drop_remote_type` *название_типа*

Аргумент	Описание
<i>название_типа</i>	Удаляемый тип сообщений. Данная строка должна содержать одно из следующих значений: <ul style="list-style-type: none">◆ file◆ ftp◆ smtp◆ mapi◆ vim

Дополнительная информация "Процедура `sp_remote_type`" на стр. 373,
"Оператор DROP REMOTE MESSAGE TYPE" на стр. 311.

Описание Удаление названного типа сообщений из базы данных.

Пример ◆ Нижеприведенный оператор удаляет из базы данных тип сообщений MAPI:

```
sp_drop_remote_type mapi
go
```

Процедура `sp_drop_sql_remote`

Назначение Удаление из базы данных системных объектов SQL Remote.

Синтаксис `sp_drop_SQL_remote`

Дополнительная информация "Процедура `sp_queue_drop`" на стр. 361.

Описание Данная процедура удаляет из базы данных системные объекты SQL Remote, которые больше не требуются в системе SQL Remote.

Единственный неудаляемый объект SQL Remote – сама процедура `sp_drop_sql_remote` (процедура не может удалить из базы данных саму себя). Для завершения удаления SQL Remote требуется явное удаление процедуры `sp_drop_sql_remote` после ее вызова.

Процедура `sp_drop_sql_remote` не удаляет из базы данных объекты очереди с сохранением. Для удаления очереди с сохранением используется процедура `sp_queue_drop` (см. раздел "Процедура `sp_queue_drop`" на стр. 418).

Пример ♦ Нижеприведенные операторы удаляют из базы данных системные объекты SQL Remote:

```
sp_drop_SQL_remote_type
go

drop procedure sp_drop_SQL_remote
go
```

Процедура *sp_grant_consolidate*

Назначение Определение базы данных, находящейся в иерархии SQL Remote на один уровень выше текущей базы данных, для получения сообщений от текущей базы данных. Эта процедура применима только к базам данных Adaptive Server Enterprise, выполняющим функцию удаленных баз данных.

Синтаксис **sp_grant_consolidate** *имя_пользователя*, *название_типа*, *адрес* [*, периодичность*] [*, время_отправки*]

Аргумент	Описание
<i>имя_пользователя</i>	Идентификатор пользователя, который будет получать сообщения SQL Remote.
<i>название_типа</i>	Тип сообщения, который будет при этом использоваться. Значением может быть одно из следующих: <ul style="list-style-type: none"> ◆ file ◆ ftp ◆ smtp ◆ mapi ◆ vim
<i>адрес</i>	Строка, содержащая адрес для указанного типа сообщений, на который нужно отправлять сообщения репликации для данного пользователя.
<i>периодичность</i>	Строка, содержащая одно из следующих значений: <ul style="list-style-type: none"> ◆ SEND EVERY. Указывает, что сообщения отправляются с периодичностью, указанной аргументом <i>время_отправки</i>. ◆ SEND AT. Указывает, что сообщения отправляются в указанное <i>время_отправки</i>.
<i>время_отправки</i>	Строка, содержащая указание времени со следующими значениями: <ul style="list-style-type: none"> ◆ Если значением аргумента <i>периодичность</i> является SEND EVERY, то <i>время_отправки</i> определяет отрезок времени между сообщениями. ◆ Если значением аргумента <i>периодичность</i> является SEND AT, то <i>время_отправки</i> определяет время суток, назначенное для отправки сообщений. Если аргумент <i>периодичность</i> не указан, Message Agent отправляет сообщения, а затем завершает свою работу.

Дополнительная информация "Процедура *sp_grant_remote*" на стр. 342,
 "Процедура *sp_revoke_consolidate*" на стр. 434,
 "Оператор GRANT CONSOLIDATE" на стр. 313.

Описание Если в системе SQL Remote сервер Adaptive Server Enterprise действует как удаленная база данных, этой базе данных, находящейся выше текущей базы

данных, необходимо с помощью процедуры **sp_grant_consolidate** предоставить полномочия CONSOLIDATE.

Консолидированный пользователь идентифицируется системой передачи сообщений, которая определяет метод отправки сообщений консолидированному пользователю и получения сообщений от консолидированного пользователя. Указанный адрес должен быть действительным адресом для системы передачи сообщений, заключенным в одиночные кавычки.

Процедура **sp_grant_consolidate** требуется для получения сообщений удаленной базой данных, но не осуществляет подписку удаленных пользователей на какие-либо данные. Для осуществления подписки на данные для идентификатора пользователя должна быть создана подписка на одну из публикаций в текущей базе данных.

Необязательный аргумент *периодичность* определяет частоту отправки сообщений. Аргумент *время_отправки* содержит значение времени, которое соответствует либо отрезку времени между отправками сообщений (в случае SEND EVERY), либо времени дня, когда производится отправка сообщений (в случае SEND AT). В случае SEND AT сообщения отправляются один раз в сутки.

Если аргумент *периодичность* не указан, Message Agent осуществляет отставку сообщений, а затем завершает свою работу. Для обеспечения непрерывной работы Message Agent необходимо указать периодичность для каждого пользователя с полномочиями REMOTE или CONSOLIDATE.

Пример

- ◆ Нижеприведенный оператор предоставляет полномочия CONSOLIDATE пользователю `hq_user` со следующими параметрами: используется система передачи сообщений путем совместного доступа к файлам, для отправки сообщений указан адрес `hq_dir`, периодичность не задана. В этом случае Message Agent будет работать в режиме пакетной передачи сообщений.

```
sp_grant_consolidate
    @user_name=hq_user,
    @address=hq_dir,
    @type_name=file
go
```

Процедура *sp_grant_remote*

Назначение Определение базы данных, находящейся в иерархии SQL Remote на один уровень ниже текущей базы данных, для получения сообщений от текущей базы данных. Такие базы данных могут упоминаться как удаленные пользователи.

Синтаксис **sp_grant_remote** *имя_пользователя, название_типа, адрес*
 [, *периодичность*] [, *время_отправки*]

Аргумент	Описание
<i>имя_пользователя</i>	Идентификатор пользователя, который будет получать сообщения SQL Remote.
<i>название_типа</i>	Тип сообщений, который будет использоваться. Тип должен быть одним из следующих: <ul style="list-style-type: none"> ◆ file ◆ ftp ◆ smtp ◆ mapi ◆ vim
<i>адрес</i>	Строка, содержащая адрес для указанного типа сообщений, на который нужно отправлять сообщения репликации для данного пользователя.
<i>периодичность</i>	Строка, содержащая одно из следующих значений: <ul style="list-style-type: none"> ◆ SEND EVERY Указывает, что сообщения отправляются с периодичностью, указанной аргументом <i>время_отправки</i>. ◆ SEND AT Указывает, что сообщения отправляются в указанное <i>время_отправки</i>.
<i>время_отправки</i>	Дополнительная строка, содержащая указание времени со следующими значениями: <ul style="list-style-type: none"> ◆ Если значением аргумента <i>периодичность</i> является SEND EVERY, то <i>время_отправки</i> определяет отрезок времени между сообщениями. ◆ Если значением аргумента <i>периодичность</i> является SEND AT, то <i>время_отправки</i> определяет время суток, назначенное для отправки сообщений. Если аргумент <i>периодичность</i> не указан, Message Agent отправляет сообщения, а затем завершает свою работу.

Дополнительная информация "Процедура *sp_revoke_remote*" на стр. 378,
 "Оператор GRANT REMOTE" на стр. 315.

Описание В системе SQL Remote каждой базе данных, получающей сообщения от текущей базы данных, с помощью процедуры **sp_grant_remote** должны быть предоставлены полномочия REMOTE.

Удаленный пользователь идентифицируется системой передачи сообщений, которая определяет отправки сообщений консолидированному пользователю и

получения сообщений от консолидированного пользователя. Указанный адрес должен быть действительным адресом для системы передачи сообщений, заключенным в одиночные кавычки.

Процедура `sp_grant_remote` требуется для получения сообщений удаленной базой данных, но не осуществляет подписку удаленных пользователей на какие-либо данные. Для осуществления подписки на данные для идентификатора пользователя должна быть создана подписка на одну из публикаций в текущей базе данных.

Необязательный аргумент *периодичность* определяет частоту отправки сообщений. Аргумент *время_отправки* содержит значение времени, которое соответствует либо отрезку времени между отправками сообщений (в случае SEND EVERY), либо времени дня, когда производится отправка сообщений (в случае SEND AT). В случае SEND AT сообщения отправляются один раз в сутки.

Если аргумент *периодичность* не указан, Message Agent осуществляет отставку сообщений, а затем завершает свою работу. Для обеспечения непрерывной работы Message Agent необходимо указать периодичность для каждого пользователя с полномочиями REMOTE.

Предполагается, что во множестве консолидированных базах данных Message Agent будет работать непрерывно, поэтому для всех удаленных баз данных аргумент *периодичность* должен быть определен. Как правило, устанавливается ежедневная отправка сообщений пользователям портативных компьютеров (SEND AT), а на удаленные сервера отправка осуществляется каждый час или два (SEND EVERY). Для повышения эффективности рекомендуется использовать как можно меньше разных значений времени отправки сообщений.

Пример

- ◆ Нижеприведенный оператор предоставляет удаленные полномочия пользователю **SamS** со следующими параметрами: используется система электронной почты MAPI, для отправки сообщений указан адрес **Singer, Samuel**, периодичность - каждые два часа:

```
exec sp_grant_remote 'SamS',  
    'mapi',  
    'Singer, Samuel',  
    'SEND EVERY',  
    '02:00'  
go
```

Процедура `sp_link_option`

Назначение	Установка параметра управления сообщениями для системы передачи сообщений SQL Remote.
Синтаксис	<code>sp_link_option</code> <i>имя-системы</i> , <i>идентификатор-пользователя</i> , <i>имя-параметра</i> , <i>значение-параметра</i>
Параметры	<p><i>имя-системы</i>: file ftp mapi smtp vim</p> <p><i>имя-параметра-системы</i>: <i>параметр-file</i> <i>параметр-ftp</i> <i>параметр-mapi</i> <i>параметр-smtp</i> <i>параметр-vim</i></p> <p><i>параметр-file</i>: debug directory</p> <p><i>параметр-ftp</i>: active_mode debug host password port root_directory user</p> <p><i>параметр-mapi</i>: debug force_download ipm_receive ipm_send profile</p> <p><i>параметр-smtp</i>: debug local_host pop3_host pop3_password pop3_userid smtp_host top_supported</p> <p><i>параметр-vim</i>: debug password path receive_all send_vim_mail userid</p> <p><i>значение-параметра-системы</i>: <i>строка</i></p>
Полномочия	Требуются полномочия администратора БД. Издатель может задать собственные параметры.
Побочные эффекты	Автоматическое подтверждение.
Дополнительная информация	"Оператор SET REMOTE OPTION" на стр. 323.
Описание	<p>Message Agent сохраняет параметры системы передачи сообщений, введенные пользователем в диалоговое окно установки системы передачи сообщений при первом использовании системы. В этом случае нет необходимости в явном выполнении данной процедуры. Данная процедура наиболее полезна при подготовке консолидированной базы данных с целью извлечения большого количества баз данных.</p> <p>Имена параметров чувствительны к регистру. Чувствительность значений параметров к регистру зависит от параметра: булевы значения нечувствительны к регистру, в то время как чувствительность к регистру для паролей, имен папок и других строк зависит от чувствительности к регистрам файловой системы (для имен каталогов) или базы данных (для идентификаторов пользователей и паролей).</p> <p>Идентификатор пользователя. Если <i>идентификатор-пользователя</i> не указан, предполагается, что это идентификатор текущего издателя.</p> <p>Значения параметров. Значения параметров зависят от конкретной системы передачи сообщений. Для получения дополнительной информации см. следующие разделы:</p> <ul style="list-style-type: none"> ◆ "Система передачи сообщений FILE" на стр. 187,

- ◆ "Система передачи сообщений FTP" на стр. 188,
- ◆ "Система передачи сообщений MAPI" на стр. 191,
- ◆ "Система передачи сообщений SMTP" на стр. 189,
- ◆ "Система передачи сообщений VIM" на стр. 192.

Пример

Нижеприведенный оператор задает *ftp.mycompany.com* в качестве FTP-хоста для ftp-канала пользователя *myuser*:

```
exec sp_link_option ftp, myuser,  
host, 'ftp.mycompany.com'
```

Процедура `sp_modify_article`

Назначение Изменение описания статьи в процедуре.

Синтаксис `sp_modify_article`
имя_публикации,
имя_таблицы,
 [*выражение_where*],
 [*выражение_subscribe_by*]
 [*представление_subscribe_by*]

Аргумент	Описание
<i>имя_публикации</i>	Имя публикации, статья которой должна быть изменена.
<i>имя_таблицы</i>	Таблица, содержащая статью.
<i>выражение_where</i>	Этот необязательный аргумент должен быть именем столбца или иметь значение NULL. Публикация включает только строки, для которых указанное имя столбца - не NULL. Значение по умолчанию - NULL, что означает, что никакие строки из публикации не исключаются.
<i>выражение_subscribe_by</i>	Новое выражение подписки, определяющее, какие строки должны быть включены в публикацию для каждой подписки. Значение по умолчанию - NULL.
<i>представление_subscribe_by</i>	Представление, определяющее столбцы и строки, которые будут включены в публикацию. Значение по умолчанию - NULL. ☞ Для получения дополнительной информации см. разделы "Повышение производительности при операции извлечения" на стр. 134 и "Повышение производительности при операции извлечения совместно используемых строк" на стр. 140.

Дополнительная информация "Процедура `sp_add_article`" на стр. 331,
 "Процедура `sp_remove_article`" на стр. 374,
 "Оператор ALTER PUBLICATION" (ALTER PUBLICATION statement) на стр. 216
 в документе "*Справочник по SQL для ASA*" (ASA SQL Reference Manual).

Описание Изменение описания статьи в публикации. Могут быть изменены: выражение WHERE, выражение подписки и представление подписки.

Как и другие изменения в определениях данных, в реальной ситуации данная процедура должна выполняться только при минимальной нагрузке системы SQL Remote.

☞ Для получения дополнительной информации о требованиях по минимальной нагрузке см. раздел "Внесение изменений в схему" на стр. 284.

Примеры Нижеприведенный оператор вносит изменения в статью публикации **SalesRepData**, которая берет информацию из таблицы **Customer**; выражение подписки при этом отсутствует:

```
sp_modify_article SalesRepData, Customer
go
```

Нижеприведенный оператор изменяет статью в публикации **SalesRepData**, которая берет информацию из таблицы **Customer**, при этом имеется выражение подписки, которое является столбцом **rep_key**:

```
sp_modify_article SalesRepData, Customer,  
                 NULL, rep_key  
go
```

Процедура `sp_modify_remote_table`

Назначение Изменение объектов разрешения конфликтов для таблицы, отмеченной для репликации SQL Remote.

Синтаксис `sp_modify_remote_table` *имя_таблицы*,
 [*имя_процедуры_разрешения*,]
 [*старое_имя_строки*],
 [*имя_строки_удаленной_БД*]

Аргумент	Описание
<i>имя_таблицы</i>	Таблица, отмеченная для репликации SQL Remote.
<i>процедура_разрешения</i>	Имя новой хранимой процедуры для выполнения действий по разрешению возникнувших конфликтов.
<i>старое_имя_строки</i>	Имя новой таблицы для хранения значений при возникновении конфликтов.
<i>имя_строки_удаленной_БД</i>	Имя новой таблицы для хранения значений удаленной базы данных при обработке вызвавшего конфликт оператора UPDATE.

Дополнительная информация "Процедура `sp_add_remote_table`" на стр. 334,
 "Процедура `sp_remove_remote_table`" на стр. 376,
 "Разрешение конфликтов" на стр. 102.

Описание Прежде чем база данных может быть включена в любые публикации SQL Remote, каждая ее таблица должна быть отмечена для репликации с помощью `sp_add_remote_table`.

Процедура `sp_modify_remote_table` позволяет изменять способ разрешения конфликтов в случае конфликтов обновления, возникающих в данной таблице.

Кроме имени таблицы, аргументами являются имена объектов, требуемых при применении пользовательских методов разрешения конфликтов. Для пользовательских методов разрешения конфликтов необходимо указать имена двух таблиц и хранимой процедуры. Процедура `sp_modify_remote_table` не проверяет существование объектов разрешения конфликтов: их можно создать как до, так и после отключения таблицы для репликации.

Эти две таблицы должны иметь те же самые столбцы и типы данных, что и таблица, указанная как *имя_таблицы*.

Пример Нижеприведенный оператор указывает SQL Remote использовать процедуру `resolve_cust` и таблицы `old_cust` и `remote_cust` для разрешения конфликтов обновления в таблице **Customer**:

```
sp_add_remote_table Customer, resolve_cust,  
old_cust, remote_cust  
go
```

Процедура sp_passthrough

Назначение Отправка оператора SQL в режиме ретрансляции.

Синтаксис **sp_passthrough** *оператор*

Аргумент	Описание
<i>оператор</i>	Строка, содержащая оператор, который будет выполнен в режиме ретрансляции.

Дополнительная информация "Процедура sp_passthrough_piece" на стр. 350,
"Процедура sp_passthrough_stop" на стр. 352,
"Процедура sp_passthrough_subscription" на стр. 353,
"Процедура sp_passthrough_user" на стр. 354,
"Оператор PASSTHROUGH" на стр. 317.

Описание Отправка операций в режиме ретрансляции. Получатели оператора в режиме ретрансляции определяются при предыдущих вызовах процедур **sp_passthrough_user** и **sp_passthrough_subscription**.

Длина строк должна быть менее 255 символов. В случаях операторов SQL, имеющих длину более 255 символов, необходимо последовательно выполнить необходимо количество вызовов процедуры **sp_passthrough_piece**, а затем вызвать **sp_passthrough** для обработки последней части оператора, после чего запустить процесс репликации.

Предостережение

Всегда проверяйте операции ретрансляции на тестовой базе данных с подписанной удаленной базой данных. Нельзя запускать непроверенные сценарии ретрансляции в действующей базе данных.

Пример

- ◆ Нижеприведенный оператор отправляет оператор CREATE TABLE текущим получателям операторов в режиме ретрансляции.

```
exec sp_passthrough
    'CREATE TABLE simple (
        id integer NOT NULL,
        name char(50) )'
go
```

Процедура *sp_passthrough_piece*

Назначение Построение длинного оператора SQL для ретрансляции.

Синтаксис *sp_passthrough_piece* строка

Аргумент	Описание
<i>строка</i>	Часть оператора, который будет выполнен в режиме ретрансляции.

Дополнительная информация "Процедура *sp_passthrough*" на стр. 349,
 "Процедура *sp_passthrough_stop*" на стр. 352,
 "Процедура *sp_passthrough_subscription*" на стр. 353,
 "Процедура *sp_passthrough_user*" на стр. 354,
 "Оператор PASSTHROUGH" на стр. 317.

Описание Процедура *sp_passthrough* (см. раздел "Процедура *sp_passthrough*" на стр. 349) используется для отправки операторов непосредственно группе удаленных пользователей. Операторы, длина которых превышает 255 символов, при отправке разбиваются на несколько частей.

Для построения и отправки длинных операторов SQL вызывается ***sp_passthrough_piece*** для всех, кроме заключительной части оператора, а затем вызывается ***sp_passthrough*** для обработки его заключительной части. Эта процедура завершает построение оператора и выполняет его репликацию.

Все части оператора ретрансляции должны быть сформированы в пределах одной транзакции.

Пример ♦ Нижеприведенные операторы отправляют длинный оператор в режиме ретрансляции по списку текущих получателей для режима ретрансляции:

```
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
```

Процедура `sp_passthrough_stop`

Назначение	Сброс режима ретрансляции.
Синтаксис	<code>sp_passthrough_stop</code>
Дополнительная информация	"Процедура <code>sp_passthrough</code> " на стр. 349, "Процедура <code>sp_passthrough_subscription</code> " на стр. 353, "Процедура <code>sp_passthrough_user</code> " на стр. 354, "Оператор <code>PASSTHROUGH</code> " на стр. 317.
Описание	Процедура <code>sp_passthrough_stop</code> очищает список получателей операторов в режиме ретрансляции и удаляет все операторы, которые формируются в текущее время.
Пример	<ul style="list-style-type: none">◆ Нижеприведенный оператор очищает список получателей операторов в режиме ретрансляции.<pre>exec sp_passthrough_stop go</pre>

Процедура `sp_passthrough_subscription`

Назначение Добавляет подписчиков на данную публикацию в список получателей операторов в режиме ретрансляции.

Синтаксис `sp_passthrough_subscription` *имя_публикации*, *значение_подписки*

Аргумент	Описание
<i>имя_публикации</i>	Имя публикации.
<i>значение_подписки</i>	Значение подписки для получателей операторов ретрансляции.

Дополнительная информация "Процедура `sp_passthrough`" на стр. 349,
 "Процедура `sp_passthrough_piece`" на стр. 350,
 "Процедура `sp_passthrough_stop`" на стр. 352,
 "Процедура `sp_passthrough_user`" на стр. 354,
 "Оператор PASSTHROUGH" на стр. 317.

Описание Данный способ является один из двух способов добавления получателей в список получения операторов в режиме ретрансляции. Другой способ заключается в том, чтобы использовать процедуру `sp_passthrough_user` (см. раздел "Процедура `sp_passthrough_user`" на стр. 354).

В результате вызова процедуры `sp_passthrough_subscription` к списку получателей добавляются все те пользователи, которые подписались на публикацию *имя_публикации* со значением подписки *значение_подписки*.

Установка по умолчанию для *значения_подписки* - NULL. В этом случае все подписчики на публикацию получают операторы ретрансляции.

Пример ♦ Нижеприведенный оператор добавляет в список получателей операторов в режиме ретрансляции подписчика или подписчиков на публикацию **SalesRepData**, которые используют значение подписки 'rep1'.

```
Sp_passthrough_subscription SalesRepData, rep1
```

Процедура *sp_passthrough_user*

Назначение Добавляет указанного пользователя в список получателей операторов в режиме ретрансляции.

Синтаксис **sp_passthrough_user** *имя_пользователя*

Аргумент	Описание
<i>имя_пользователя</i>	Пользователь, который будет добавлен в список получателей.

Дополнительная информация "Процедура *sp_passthrough*" на стр. 349,
 "Процедура *sp_passthrough_piece*", на стр. 407,
 "Процедура *sp_passthrough_stop*" на стр. 352,
 "Процедура *sp_passthrough_subscription*" на стр. 353,
 "Оператор PASSTHROUGH" на стр. 317.

Описание Данный способ является один из двух способов добавления получателей в список получения операторов в режиме ретрансляции. Другой способ заключается в том, чтобы использовать процедуру *sp_passthrough_subscription* (см. раздел "Процедура *sp_passthrough_subscription*" на стр. 353).

Процедура *sp_passthrough_user* добавляет указанного пользователя в список получателей операторов в режиме ретрансляции. Список остается в силе до сброса, который выполняется процедурой *sp_passthrough_stop* (см. раздел "Процедура *sp_passthrough_stop*" на стр. 352).

Пример ♦ Нижеприведенный оператор добавляет пользователя *field_user* в список получателей операторов в режиме ретрансляции:

```
sp_passthrough_user 'field_user'
go
```

Процедура `sp_populate_sql_anywhere`

Назначение Создание копии системных таблиц Adaptive Server Anywhere в TEMPDB. Данная процедура используется утилитой извлечения *ssxtract*.

Синтаксис **`sp_populate_SQL_anywhere`**

Описание Создание набора системных таблиц Adaptive Server Anywhere для удаленной базы данных Adaptive Server Anywhere в TEMPDB. Эта информация используется утилитой извлечения для создания схемы базы данных Adaptive Server Anywhere из набора публикаций в консолидированной базе данных Adaptive Server Enterprise.

Данная процедура используется утилитой извлечения *ssxtract*. Вызывать ее непосредственно нельзя.

Процедура `sp_publisher`

Назначение Задание или удаление издателя текущей базы данных.

Синтаксис `sp_publisher [имя_пользователя]`

Аргумент	Описание
<i>имя_пользователя</i>	Идентификатор пользователя, который будет являться идентификатором издателя для этой базы данных.

Дополнительная информация "Управление полномочиями SQL Remote" на стр. 175,
"Оператор GRANT PUBLISH" на стр. 314.

Описание В системе SQL идентификация каждой базы данных в исходящих сообщениях Remote производится по идентификатору пользователя, который является **издателем**. Процедура `sp_publisher` задает идентификатор издателя, указываемый в исходящих сообщениях .

Каждая база данных может иметь не более одного издателя; если издатель уже существует, `sp_publisher` изменяет имя издателя.

Если аргумент *имя_пользователя* не указан, текущий издатель удаляется, и для базы данных устанавливается состояние отсутствия издателя. При этом удаляются только полномочия издателя; идентификатор пользователя из базы данных не удаляется.

Примеры

- ◆ Нижеприведенный оператор определяет пользователя **joe** в качестве издателя текущей базы данных:

```
sp_publisher joe
go
```

- ◆ Нижеприведенный оператор задает отсутствие издателей у текущей базы данных:

```
sp_publisher
go
```

Процедура `sp_queue_clean`

Назначение	Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно.
Синтаксис	<code>sp_queue_clean</code>
Описание	Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно. Она удаляет из очереди с сохранением любые транзакции, которые завершились после того, как началось выполнение самой старой незавершенной транзакции при предыдущем сканировании журнала.

Процедура sp_queue_confirmed_delete_old

Назначение Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно.

Синтаксис **sp_queue_confirmed_delete_old**

Описание Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно. Она удаляет из очереди с сохранением любые транзакции, смещения которых выводятся в таблице **sr_confirmed_transaction**.

Процедура `sp_queue_confirmed_transaction`

Назначение	Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно.
Синтаксис	<code>sp_queue_confirmed_transaction</code> <i>смещение</i>
Описание	Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно. Она добавляет полученное смещение в таблицу <code>sr_confirmed_transaction</code> . SQL Remote удаляет из очереди с сохранением любые транзакции, смещения которых соответствуют данному смещению.

Процедура sp_queue_delete_old

Назначение	Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно.
Синтаксис	sp_queue_delete_old
Описание	Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно. Она удаляет из очереди с сохранением любые транзакции, которые были подтверждены всеми удаленными базами данных.

Процедура `sp_queue_drop`

Назначение	Удаление объектов очереди с сохранением из базы данных.
Синтаксис	<code>sp_queue_drop</code>
Дополнительная информация	"Процедура <code>sp_drop_sql_remote</code> " на стр. 339.
Описание	<p>Удаление системных объектов очереди с сохранением из базы данных, после чего база данных перестает поддерживать очередь с сохранением SQL Remote.</p> <p>При этом единственный неудаляемый объект очереди с сохранением – сама процедура <code>sp_queue_drop</code> (процедура не может удалить из базы данных саму себя). Для завершения удаления очереди с сохранением требуется явно удалить процедуру <code>sp_queue_drop</code> после вызова данной процедуры.</p> <p>Процедура <code>sp_queue_drop</code> не удаляет системные объекты SQL Remote из базы данных. Для удаления системных объектов SQL Remote используется процедура <code>sp_drop_sql_remote</code> (см. раздел "Процедура <code>sp_drop_sql_remote</code>" на стр. 339).</p>
Примеры	<ul style="list-style-type: none">◆ Нижеприведенные операторы удаляют из базы данных объекты очереди с сохранением:<pre>sp_queue_drop go drop procedure sp_queue_drop go</pre>

Процедура `sp_queue_dump_database`

Назначение	Используется при восстановлении после отказа носителей в случаях, когда очередь с сохранением находится в отдельной (без объектов SQL Remote) базе данных.
Синтаксис	<code>sp_queue_dump_database</code>
Дополнительная информация	"Процедура <code>sp_queue_dump_transaction</code> " на стр. 363, "Особенности восстановления очереди с сохранением" на стр. 282.
Описание	<p>Хранение очереди с сохранением в отдельной базе данных усложняет резервное копирование и восстановление, поскольку необходимо восстанавливать согласованные версии двух баз данных.</p> <p>При нормальном восстановлении эти две базы данных автоматически становятся согласованными, хотя восстановление при отказе носителя требует определенной осторожности в действиях. При восстановлении дампов базы данных важно восстановить очередь с сохранением до точки согласования. Процедура <code>sp_queue_dump_database</code> упрощает процесс восстановления после отказа носителей. Она вызывается при каждом сканировании дампов баз данных.</p> <p>По умолчанию данная процедура не выполняет никаких операций. Однако ее можно изменить так, чтобы она выполняла команду дампа базы данных для базы данных очереди с сохранением.</p>

Процедура `sp_queue_dump_transaction`

Назначение	Используется при восстановлении после отказа носителей в случаях, когда очередь с сохранением находится в отдельной (без объектов SQL Remote) базе данных.
Синтаксис	<code>sp_queue_dump_transaction</code>
Дополнительная информация	"Процедура <code>sp_queue_dump_database</code> " на стр. 362, "Особенности восстановления очереди с сохранением" на стр. 282.
Описание	<p>Хранение очереди с сохранением в отдельной базе данных усложняет резервное копирование и восстановление, поскольку необходимо восстанавливать согласованные версии двух баз данных.</p> <p>При нормальном восстановлении эти две базы данных автоматически становятся согласованными, хотя восстановление при отказе носителя требует определенной осторожности в действиях. При восстановлении дампов базы данных важно восстановить очередь с сохранением до точки согласования. Процедура <code>sp_queue_dump_transaction</code> упрощает процесс восстановления после отказа носителей. Она вызывается при каждом сканировании дампов транзакций.</p> <p>По умолчанию данная процедура не выполняет никаких операций. Однако ее можно изменить так, чтобы она выполняла команду дампа транзакций для базы данных очереди с сохранением.</p>

Процедура sp_queue_get_state

Назначение Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно.

Синтаксис **sp_queue_get_state**

Описание Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно. Она возвращает описание текущего состояния очереди с сохранением.

Процедура `sp_queue_log_transfer_reset`

Назначение	Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно.
Синтаксис	<code>sp_queue_log_transfer_reset</code>
Описание	Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно. Она устанавливает идентификаторы страницы и строки таблицы <code>sr_queue_state</code> в ноль.

Процедура sp_queue_read

Назначение	Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно.
Синтаксис	sp_queue_read <i>начальное_смещение, конечное_смещение</i>
Описание	Данная процедура считывает транзакции из очереди с сохранением. Она предназначена исключительно для использования в Message Agent.

Процедура `sp_queue_reset`

Назначение	Сброс сервера до момента, когда очередь с сохранением пуста.
Синтаксис	<code>sp_queue_reset</code>
Описание	<p>Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно в реальной ситуации. Она удаляет все строки из таблицы <code>sr_transaction</code> очереди с сохранением и сбрасывает таблицу <code>sr_queue_state</code> новой системы SQL Remote.</p> <p>Данная процедура может использоваться для сброса сервера на этапе разработки.</p>

Процедура sp_queue_set_confirm

Назначение	Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно.
Синтаксис	sp_queue_set_confirm <i>смещение_подтверждения</i>
Описание	Данная процедура используется SQL Remote Message Agent и не должна может быть вызвана непосредственно. Она задает минимальное смещение подтверждения от всех удаленных пользователей в таблице sr_queue_state .

Процедура `sp_queue_set_progress`

Назначение	Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно.
Синтаксис	<code>sp_queue_set_progress</code> <i>идентификатор_страницы</i> , <i>идентификатор_строки</i> , <i>смещение_подтверждения</i> , <i>смещение_резервного_копирования</i> , <i>маркер</i>
Описание	Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно. Она задает значение индикатора хода сканирования журнала транзакций в таблице <code>sr_queue_state</code> .

Процедура sp_queue_transaction

Назначение	Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно.
Синтаксис	sp_queue_transaction <i>смещение, идентификатор_пользователя</i>
Описание	Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно. Она добавляет новую транзакцию в очередь с сохранением.

Процедура `sp_remote`

Назначение Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно, с единственным исключением, описанным ниже. Данная процедура управляет строками в таблице `sr_remoteuser`.

Синтаксис `sp_remote операция, имя_пользователя [, смещение] [, подтверждение]`

Аргумент	Описание
<i>операция</i>	Название действия. Единственное значение, которое должно использоваться пользователем, - reset (сброс); все другие значения предназначены для использования Message Agent.
<i>имя_пользователя</i>	Сбрасываемое имя удаленного пользователя.
<i>смещение</i>	Не используется.
<i>подтверждение</i>	Не используется.

Описание Данная процедура используется SQL Remote Message Agent и не должна вызываться непосредственно, с единственным исключением, которым является вызов операции **reset** (сброс). Эта процедура используется для управления результатами отслеживания сообщений в таблице `sr_remoteuser`.

В одном особом случае процедура может вызываться непосредственно - при создании пользовательского процесса извлечения базы данных:

```
sp_remote reset, remote_user
```

где *remote_user* - имя удаленного пользователя.

Данная команда активизирует все подписки для удаленного пользователя в одной транзакции. Она задает значения **log_sent** и **confirm_sent** в таблице `sr_remoteuser` в соответствии с текущей позицией в журнале транзакций. Она также задает созданные и активизированные значения в **sr_subscription** в соответствии с текущей позицией в журнале транзакций для всех подписок данного удаленного пользователя. Процедура не выполняет подтверждения. После ее вызова необходимо сделать явное подтверждение.

Для безопасной записи процесса извлечения в действующей БД данные должны извлекаться на третьем уровне изоляции в той же самой транзакции, которая активизирует подписки.

Процедура `sp_remote_option`

Назначение Задание параметра SQL Remote.

Синтаксис `sp_remote_option имя_параметра, значение_параметра`

Аргумент	Описание
<i>имя_параметра</i>	Имя одного из параметров SQL Remote.
<i>значение_параметра</i>	Значение, установленное для данного параметра.

Дополнительная информация "Параметры SQL Remote" на стр. 272.

Описание Параметры SQL Remote обеспечивают управление поведением системы репликации. В Adaptive Server Enterprise доступны следующие параметры:

ПАРАМЕТРЫ	ЗНАЧЕНИЯ	ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ
Blob_threshold	Целое число, в Кб	256
Compression	От -1 до 9	6
Delete_old_logs	ON, OFF	OFF
Qualify_owners	ON, OFF	OFF
Quote_all_identifiers	ON, OFF	OFF
Replication_error	<i>имя-процедуры</i>	NULL
SR_Date_Format	<i>строка-времени</i>	hh:nn:ss.Ssssss
SR_Time_Format	<i>строка-даты</i>	yyyy/mm/dd
SR_Timestamp_Format	<i>строка-метки-времени</i>	yyyy/mm/dd hh:nn:ss.Ssssss
Subscribe_by_remote	ON, OFF	ON
Verify_threshold	<i>целое число</i>	256
Verify_all_columns	ON, OFF	OFF

☞ Полное описание этих параметров см. в разделе "Параметры SQL Remote" на стр. 272.

Пример

- ◆ Нижеприведенный оператор устанавливает параметр `Verify_all_columns` в значение OFF для отключения автоматической проверки старых значений операторов обновления, выполненных в Message Agent, для всех столбцов.

```
sp_remote_option Verify_all_columns, OFF
go
```

Процедура `sp_remote_type`

Назначение Создание или изменение типа сообщений SQL Remote.

Синтаксис `sp_remote_type` *название_типа* *адрес_издателя*

Аргумент	Описание
<i>название_типа</i>	Название создаваемого или изменяемого типа сообщений. Значением может быть одно из следующих: <ul style="list-style-type: none"> ◆ file ◆ ftp ◆ smtp ◆ mapi ◆ vim
<i>адрес_издателя</i>	Адрес издателя для этого типа сообщений.

Дополнительная информация "Процедура `sp_drop_remote_type`" на стр. 338,
"Оператор ALTER REMOTE MESSAGE TYPE" на стр. 304.

Описание Message Agent производит отправку исходящих сообщений из базы данных с использованием одной из поддерживаемых систем передачи сообщений. Обратные сообщения пользователей, использующих указанную систему, отправляются на указанный адрес при условии, что удаленная база данных создавалась утилитой извлечения. Message Agent активизирует системы только при условии наличия удаленных пользователей для этих систем.

Адресом является адрес издателя для указанной системы передачи сообщений. Если такой системой является электронная почта, строкой адреса должен быть действительный адрес электронной почты. Если такой системой является система совместного доступа к файлам, строкой адреса должен быть подпапка в наборе папок, указанная в переменной среды SQLREMOTE или записи реестра. При отсутствии задания строки адреса в качестве адреса назначается текущая папка.

Пример В следующем примере для базы данных создается тип сообщений FILE, и адрес издателя задается как подпапка местоположения SQLREMOTE с именем *publisher*:

```
sp_remote_type file, publisher
go
```

Процедура `sp_remove_article`

Назначение Удаление статьи из публикации.

Синтаксис `sp_remove_article` *имя_публикации*, *имя_таблицы*

Аргумент	Описание
<i>имя_публикации</i>	Имя публикации, из которой должна быть удалена статья.
<i>имя_таблицы</i>	Таблица, содержащая статью.

Дополнительная информация "Процедура `sp_add_article`" на стр. 331,
"Оператор ALTER PUBLICATION" (ALTER PUBLICATION statement) на стр. 216
в документе "*Справочник по SQL для ASA*" (ASA SQL Reference Manual).

Описание Процедура `sp_add_article` удаляет статью из публикации. При этом из публикации может удаляться любая статья, включая части названной таблицы.

Пример ♦ Нижеприведенный оператор удаляет все статьи, использующие часть таблицы **SalesRep** из публикации с именем **SalesRepData**:

```
sp_remove_article SalesRepData, SalesRep
go
```

Процедура `sp_remove_article_col`

Назначение Удаление столбца из статьи в публикации.

Синтаксис `sp_remove_article_col имя_публикации, имя_статьи, имя_столбца`

Аргумент	Описание
<code>имя_публикации</code>	Имя публикации, которой принадлежит статья.
<code>имя_статьи</code>	Статья, из которой должен быть удален столбец.
<code>имя_столбца</code>	Столбец, который будет удален из статьи.

Дополнительная информация "Процедура `sp_add_article_col`" на стр. 333,
 "Процедура `sp_remove_article`" на стр. 374,
 "Оператор ALTER PUBLICATION" (ALTER PUBLICATION statement) на стр. 216
 в документе "*Справочник по SQL для ASA*" (ASA SQL Reference Manual).

Описание Процедура `sp_remove_article_col` позволяет удалить столбец из публикации.

Для удаления столбца с помощью `sp_remove_article_col` этот столбец должен быть уже явно добавлен к публикации. Для этого используется процедура `sp_add_article_col` (см. раздел "Процедура `sp_add_article_col`" на стр. 333). Хотя процедура `sp_add_article` (см. раздел "Процедура `sp_add_article`" на стр. 331) добавляет все столбцы таблицы к публикации без использования `sp_add_article_col`, удаление отдельных столбцов из такой публикации при использовании `sp_remove_article_col` невозможно.

Пример ♦ Нижеприведенный оператор удаляет столбец `emp_lname` таблицы `employee` из публикации с именем `Personnel`:

```
sp_remove_article_col 'Personnel', 'employee',
'emp_lname'
go
```

Процедура `sp_remove_remote_table`

Назначение Отметка таблицы как недоступной для репликации SQL Remote.

Синтаксис `sp_remove_remote_table` *имя_таблицы*

Аргумент	Описание
<i>имя_таблицы</i>	Таблица, которая будет отмечена как недоступная для репликации SQL Remote.

Дополнительная информация "Процедура `sp_add_remote_table`" на стр. 334,
"Процедура `sp_modify_remote_table`" на стр. 348.

Описание Эта процедура отмечает таблицу как недоступную для репликации. После вызова этой процедуры данные в таблице не будут совместно использоваться быть другими базами данных в системе SQL Remote.

Пример ♦ Приведенный ниже оператор отмечает таблицу **employee** как недоступную для репликации:

```
sp_remove_remote_table employee
go
```


Процедура `sp_revoke_consolidate`

Назначение Запрет получения пользователем сообщений SQL Remote от этой базы данных.

Синтаксис `sp_revoke_consolidate` *имя_пользователя*

Аргумент	Описание
<i>имя_пользователя</i>	Идентификатор пользователя, который больше не сможет выполнять функции консолидированного пользователя.

Дополнительная информация "Процедура `sp_grant_consolidate`" на стр. 340.

Описание Процедура `sp_revoke_consolidate` удаляет идентификатор пользователя консолидированной базы данных из списка пользователей, получающих сообщения от текущей базы данных.

Пример ♦ Приведенный ниже оператор отменяет полномочия консолидированного пользователя `hq_user`:

```
sp_revoke_consolidate hq_user
go
```

Процедура `sp_revoke_remote`

Назначение Запрет получения пользователем сообщений SQL Remote от этой базы данных.

Синтаксис `sp_revoke_remote` *имя_пользователя*

Аргумент	Описание
<i>имя_пользователя</i>	Идентификатор пользователя, который больше не сможет получать сообщения SQL Remote.

Дополнительная информация "Процедура `sp_grant_remote`" на стр. 342.

Описание Процедура `sp_revoke_remote` удаляет идентификатор пользователя из списка пользователей, получающих сообщения от текущей базы данных.

Пример ♦ Приведенный ниже оператор отменяет полномочия удаленного пользователя **Field User**:

```
sp_revoke_remote 'Field user'  
go
```

Процедура `sp_subscription`

Назначение Управление подписками.

Синтаксис `sp_subscription` операция,
имя_публикации,
имя_пользователя,
[значение_подписки]

Аргумент	Описание
<i>операция</i>	Операция, которая должна быть выполнена. Допустима одна из следующих операций: <ul style="list-style-type: none"> ◆ create Создание подписки пользователя на данную публикацию. ◆ drop Удаление подписки пользователя на данную публикацию. ◆ start Активизация подписки на указанную публикацию. ◆ stop Деактивизация подписки на указанную публикацию. ◆ synchronize Синхронизация подписки на указанную публикацию.
<i>имя_публикации</i>	Имя публикации, к которой относится подписка.
<i>имя_пользователя</i>	Идентификатор пользователя, подписанного на публикацию.
<i>значение_подписки</i>	Значение подписки.

Дополнительная информация "Создание подписок" на стр. 118.

Описание Процедура `sp_subscription` используется для управления подписками. Первый аргумент процедуры (*операция*) определяет выполняемое действие: создание, удаление, активизация, деактивизация или синхронизация.

Как правило, активизация и синхронизация подписки выполняются при помощи утилиты извлечения.

Пример ◆ Приведенный ниже оператор создает подписку для пользователя `SalesRep1` на публикацию `SalesRepData`, которая не имеет выражения подписки.

```
sp_subscription create,  
SalesRepData,  
SalesRep1  
go
```

Процедура `sp_subscription_reset`

Назначение	Сброс всей информации SQL Remote для всех удаленных пользователей.
Синтаксис	<code>sp_subscription_reset</code>
Описание	Данная процедура сбрасывает все записи в таблицах <code>sr_remote_user</code> и <code>sr_subscription</code> в нуль или NULL.

Приложение

Приложение содержит дополнительную информацию, которая не всегда требуется при ежедневной работе с программным продуктом.



ПРИЛОЖЕНИЕ А

Различия между SQL Remote для Adaptive Server Enterprise и SQL Remote для Adaptive Server Anywhere

Об этом приложении В данном приложении кратко описываются основные различия между SQL Remote для Adaptive Server Enterprise и SQL Remote для Adaptive Server Anywhere. Приложение содержит базовую информацию о различиях между двумя версиями данного программного продукта.

Содержание

Раздел	Страница
Типы различий	384
Различия в функциональных возможностях	385
Различия в подходах	386
Ограничения при репликации Enterprise-Enterprise	388

Типы различий

Между версиями программного обеспечения имеются следующие типы различий:

- ◆ **Функциональные возможности.** Задачи, которые могут быть выполнены в одной из этих двух версий, но не могут быть выполнены в другой.
- ◆ **Подходы.** Несмотря на схожесть получаемых результатов, в каждой версии реализован свой подход к выполнению операций. Это относится, в частности, к задачам, которые выполняются внешне различными способами, но при этом имеют один и тот же результат.
- ◆ **Различия на уровне сервера.** Задачи, связанные с SQL Remote, например, управление резервным копированием, различаются в зависимости от типа используемого сервера. Такие различия здесь не рассматриваются.

Предметом данного приложения является только репликация, при которой Adaptive Server Anywhere используется в качестве удаленной базы данных. При использовании Adaptive Server Enterprise в качестве удаленного сервера имеются дополнительные ограничения.

Различия в функциональных возможностях

Главные различия в функциональных возможностях между SQL Remote для Adaptive Server Enterprise (SRE) и SQL Remote для Adaptive Server Anywhere (SRA) следующие:

- ◆ **Изменения схем.** В SRE изменения схем должны выполняться в системе с **минимальной нагрузкой**. Под состоянием минимальной нагрузки понимается следующее:
 - ◆ **Никакие транзакции не реплицируются.** Транзакции, модифицирующие таблицы, которые должны быть изменены, не существуют. Все транзакции, модифицирующие изменяемые таблицы, необходимо отсканировать из журнала транзакций в очередь с сохранением перед тем, как будет изменена схема. Это выполняется обычным запуском Message Agent, либо при помощи параметров -i, -b. После завершения работы Message Agent в схему можно вносить изменения.
 - ◆ **Завершение работы Message Agent.** Во время внесения изменений в схему Message Agent необходимо закрыть.
 - ◆ **SQL Remote Open Server.** При работе с SQL Remote Open Server его необходимо закрыть во время внесения изменений в схему.
- ◆ **Репликация действий триггеров.** В SRE действия триггеров реплицируются. В SRA имеется возможность выполнения репликации действий триггеров, но по умолчанию они не реплицируются. При выполнении репликации действий триггеров в SRE необходимо удостовериться, что триггеры в удаленных базах данных не запускаются.
- ◆ **Доступность платформ.** Различие между двумя серверами по совместимости с различными платформами состоит в том, что SRA может работать на большем количестве различных платформ, чем SRE.
- ◆ **Определения публикаций.** Публикации в SRA могут быть иметь более высокую степень выборочности, в SRE. Например, в SRA можно использовать раздел WHERE с любыми значениями. В SRE в разделе WHERE можно использовать только условия IS NULL и IS NOT NULL.

Различия в подходах

Для реализации некоторых функций SQL Remote в SRE и SRA требуются различные подходы.

- ◆ **Разделение таблиц, не содержащих выражений подписки.** Публикации в SRA могут содержать подзапросы, которые позволяют даже таблицам, не содержащим выражений разделения, должным образом распространяться среди подписчиков. В SRE к таким таблицам должен добавляться дополнительный столбец, содержащий список подписчиков. Для поддержки этого столбца должны быть написаны соответствующие триггеры. Максимальный размер этого столбца - 255.

☞ Для получения подробной информации см. разделы "Разделение таблиц, не содержащих выражение подписки" на стр. 103 и "Разделение таблиц, не содержащих столбца подписки" на стр. 129.

- ◆ **Разрешение конфликтов.** Разрешение конфликтов в SRA выполняется при использовании специального синтаксиса триггеров. В SRE для выполнения этой задачи должны быть написаны хранимые процедуры.

☞ Для получения подробной информации см. разделы "Управление конфликтами" на стр. 143 и на стр. 166.

- ◆ **Сохранение сообщений перед отправкой.** В SRE для сохранения изменений перед репликацией применяется отдельная таблица **очереди с сохранением**. В SRA очередей с сохранением не существует. Вместо этого сообщения извлекаются из файлов текущих и старых журналов транзакций.
- ◆ **Команды.** Задачи SQL Remote (например, создание публикаций) выполняются в SRA с помощью операторов SQL, тогда как в SRE - с помощью системных хранимых процедур.

Процедуры Adaptive Server Enterprise и операторы Adaptive Server Anywhere

В SQL Remote для Adaptive Server Anywhere операторы SQL используются для выполнения задач, которые в случае Adaptive Server Enterprise выполняют хранимые процедуры. В следующей таблице перечислены процедуры SQL Remote с указанием их соответствия операторам SQL в Adaptive Server Anywhere:

Процедура Adaptive Server Enterprise	Соответствующий оператор Adaptive Server Anywhere
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

Процедура Adaptive Server Enterprise	Соответствующий оператор Adaptive Server Anywhere
<code>sp_add_article</code>	
<code>sp_add_article_col</code>	
<code>sp_add_article</code>	ALTER PUBLICATION
<code>sp_remove_article</code>	
<code>sp_add_article_col</code>	
<code>sp_remove_article_col</code>	
<code>sp_drop_publication</code>	DROP PUBLICATION
<code>sp_subscription 'create'</code>	CREATE SUBSCRIPTION
<code>sp_subscription 'drop'</code>	DROP SUBSCRIPTION
<code>sp_subscription 'start'</code>	START SUBSCRIPTION
<code>sp_subscription 'stop'</code>	STOP SUBSCRIPTION
<code>sp_subscription 'synchronize'</code>	SYNCHRONIZE SUBSCRIPTION
<code>sp_passthrough_user</code>	PASSTHROUGH FOR USERID
<code>sp_passthrough_subscription</code>	PASSTHROUGH FOR SUBSCRIPTION
<code>sp_passthrough_stop</code>	PASSTHROUGH STOP

Ограничения при репликации Enterprise-Enterprise

При использовании SQL Remote для репликации между базами данных Adaptive Server Enterprise, а не с удаленными базами данных Adaptive Server Anywhere, необходимо иметь в виду следующие ограничения:

- ◆ **Извлечение баз данных.** Сценарии RELOAD.SQL и файлы данных для построения удаленных баз данных Adaptive Server Anywhere создаются утилитой извлечения. Для настройки удаленных баз данных ASE требуется пользовательский метод извлечения.

☞ Для получения дополнительной информации о том, как создавать метод извлечения, см. раздел "Процедура sp_remote" на стр. 371.

- ◆ **Ошибки ссылочной целостности.** В Adaptive Server Enterprise ссылочная целостность всегда проверяется немедленно, в то время как в Adaptive Server Anywhere имеется параметр WAIT_FOR_COMMIT для управления временем проверки целостности. Это представляет определенные трудности при перемещении строк между удаленными базами данных, что имеет место при перераспределении областей.

Например, предположим, что таблица **Order** имеет внешний ключ к таблице **Customer**, которая, в свою очередь, имеет внешний ключ к таблице **SalesRep**. Пользователь **SalesRep** подписан на таблицу **Customer**. **SalesRep** подписан также на таблицу **Order** (эта таблица содержит избыточный столбец, поддерживаемый триггером).

При обновлении строки в таблице **Customer** для указания на новую таблицу **SalesRep** запускается триггер, чтобы обновить столбец **SalesRep** в таблице **Order**. Обновление таблицы **Customer** реплицируется как операция удаления старой **Rep** и вставка новой **Rep**. Аналогично, запущенное обновление таблицы **Order** реплицируется как операция удаления старой **Rep** и вставка новой **Rep**.

Проблема заключается в том, что SQL Remote реплицирует операции в той последовательности, в которой они происходят, а это означает, что строка таблицы **Customer** будет удалена до удаления строк таблицы **Order**. В результате возникает ошибка ссылочной целостности.

- ◆ **Обновление схем.** Если консолидированные и удаленные базы данных являются базами данных Adaptive Server Enterprise, управление обновлением схем может быть затруднено. Ретрансляцию в удаленные базы данных Adaptive Server Enterprise осуществить сложно.

Проблема заключается в необходимости обеспечения минимальной нагрузки при обновлении схем (см. раздел "Различия в функциональных возможностях" на стр. 385). При ретрансляции операторы обновления схемы помещаются в обычный поток сообщений. Операции, которые предшествуют обновлению схемы (в том же сообщении либо в предыдущем сообщении), могут оказаться не просканированными из журнала транзакций в очередь с сохранением до выполнения изменений схемы.

- ◆ **Синхронизация подписки.** Не реализуется для удаленных баз данных Adaptive Server Enterprise.

ПРИЛОЖЕНИЕ В

Поддерживаемые платформы и системы передачи сообщений

Об этом приложении В данном приложении перечислены платформы и системах передачи сообщений, поддерживаемые в системах SQL Remote.

Содержание

Раздел	Страница
Поддерживаемые системы передачи сообщений	390
Поддерживаемые операционные системы	391

Поддерживаемые системы передачи сообщений

SQL Remote производит обмен сообщениями между базами данных с использованием определенной системы передачи сообщений. SQL Remote поддерживает следующие системы передачи сообщений:

- ◆ **Система передачи сообщений путем совместного доступа к файлам** - несложная система, не требующая дополнительного программного обеспечения.
- ◆ **FTP** – передача сообщений по протоколу передачи файлов Интернет.
- ◆ **SMTP/POP** – передача сообщений по протоколу электронной почты Интернет.
- ◆ **MAPI** – интерфейс прикладного программирования для электронной почты, применяемый программных продуктах Microsoft, а также в более поздних версиях 8 или более поздних.
- ◆ **VIM** - система передачи сообщений между ПО независимых поставщиков, используемая в Lotus Notes и в некоторых версиях Lotus cc:Mail.

Не каждая из указанных систем поддерживается на всех операционных системах. Для любой из этих систем, кроме системы передачи сообщений путем совместного доступа к файлам, должно быть приобретено и установлено соответствующее программное обеспечение, позволяющее использовать данную систему при репликации SQL Remote. Пакет SQL Remote не включает программное обеспечение для используемых систем передачи сообщений.

Поддерживаемые операционные системы

SQL Remote для Adaptive Server Enterprise

SQL Remote для Adaptive Server Enterprise совместим со следующими операционными системами и системами передачи сообщений:

- ◆ **Windows NT/2000/XP** - все протоколы передачи сообщений.
- ◆ **Sun Microsystems Solaris/Sparc** - только совместный доступ к файлам, FTP и SMTP/POP.

SQL Remote для Adaptive Server Anywhere

SQL Remote для Adaptive Server Anywhere совместим со следующими операционными системами и системами передачи сообщений:

- ◆ **Windows 95/98/Me** - все системы передачи сообщений.
- ◆ **Windows NT/2000/XP** - все системы передачи сообщений.
- ◆ **Windows CE** - системы FILE и SMTP/POP. В случае системы передачи файлов *dbremote* выполняет поиск в каталоге *My Documents\Synchronized Files*. На настольном компьютере для переменной среды SQLREMOTE или параметра системы передачи сообщений *directory* для системы FILE должны быть установлены следующие значения:

```
%SystemRoot%\Profiles\идентификатор-пользователя\Personal\имя-машины-ce\Synchronized Files
```

где *идентификатор-пользователя* – имя пользователя, а *имя-машины-ce* – имя компьютера с CE; для этих параметров должны быть установлены соответствующие значения. При такой установке ActiveSync автоматически синхронизирует файлы сообщений между настольной системой и системой CE.

Проверьте параметры меню Mobile Device>Tools>ActiveSync, чтобы убедиться в том, что синхронизация файлов активизирована.

☞ Для получения информации относительно установки параметров системы передачи сообщений см. раздел "Система передачи сообщений FILE" на стр. 187.

- ◆ **Sun Microsystems Solaris/Sparc** - только совместный доступ к файлам, FTP и SMTP/POP.
- ◆ **Novell NetWare** - только совместный доступ к файлам, FTP и SMTP/POP.
- ◆ **Linux** - только совместный доступ к файлам, FTP и SMTP/POP.

Для получения подробных сведений о поддерживаемых версиях операционной системы UNIX см. документ "Подготовка к работе с SQL Anywhere Studio" (SQL Anywhere Studio Read Me First) для UNIX.

Индекс

A

ActiveSync
Windows CE, 393

Adaptive Server Anywhere
создание совместимой с Enterprise базы данных, 64

Adaptive Server Enterprise
установка и настройка SQL Remote, 19
учебный раздел по SQL Remote, 47

B

BLOB-объекты
репликация, 65, 271

C

ccMail
SQL Remote, 183

compatibility
Adaptive Server Enterprise and Adaptive Server Anywhere, 64

conflicts
пример, 145

CURRENT PUBLISHER
таблица для, 290
учебный раздел, 31, 36

CURRENT REMOTE USER
разрешение конфликтов, 145
специальная константа, 103
таблица для, 290

D

dbcc settrunc
использование, 238

dbremote, 276
безопасность, 210
введение, 8
команда, 256
описание, 193, 256
таблицы #hook_dict, 276

учебный раздел, 57
учебный раздел, 42

dsi_num_threads
параметр Replication Server, 248

F

foreign keys
перераспределение областей, 90

ftp
параметры управления, 186
тип сообщений, 183
тип сообщений, 186
устранение неисправностей, 187

L

LONG BINARY
репликация, 65

LONG VARCHAR
репликация, 65

Lotus Notes
SQL Remote, 183, 190

LTM
SQL Remote, 230

M

MAPI
параметры управления, 189
тип сообщений, 183

Message Agent
администрирование SQL Remote, 174
безопасность, 194, 210, 234
введение, 8
вывод, 210, 234
доставка сообщений, 203, 204, 206
запросы на повторную передачу, 198
запуск, 210
запуск как службы, 210
изменения схем, 238
коды пользователей, 234
команда, 255
настройки, 194
обработка подписки, 71

описание, 193, 234
 опросы, 198
 отслеживание сообщений, 203, 204, 206
 параметр -l, 194
 параметр -m, 197
 параметр -rd, 198
 параметр -gr, 198
 параметр -u, 194
 подключения, 193
 потоки, 196
 производительность, 196, 200
 рабочие потоки, 196
 режим непрерывной передачи, 193
 режим пакетной передачи, 193
 репликация триггеров, 65
 управление журналом транзакций, 214, 218, 221, 236
 учебный раздел, 42, 57

Microsoft Exchange
 профиль, 189

N

NetWare
 SQL Remote, 186
 поддерживаемые типы сообщений, 183

Notes
 SQL Remote, 183

Novell NetWare
 совместимость, 393

R

Replication Agent
 SQL Remote, 230, 243, 244

Replication Server
 SQL Remote, 230, 243, 248
 ssqueue, 248
 архитектура SQL Remote, 244
 использование с SQL Remote, 270
 конфигурирование, 248
 повторное установление соединения, 250

rs_dumpdb
 использование, 250

rs_dumptran
 использование, 250

S

salespub.sql
 пример публикации, 44

SEND AT

установка периодичности, 178

SEND EVERY
 установка периодичности, 178

SMTP
 SQL Remote, 188
 параметры управления, 188
 тип сообщений, 188
 типы сообщений, 183
 электронная почта, 188

SMTP/POP
 адреса, 189

sp_user_extraction_hook
 пример, 138

SQL Remote
 dbremote (учебный раздел), 42, 57
 Message Agent (учебный раздел), 42, 57
 ssremote (учебный раздел), 57
 TEMPDB, 19
 администрирование, 10, 225
 введение в Message Agent, 8
 выгрузка баз данных, 222
 деинсталляция, 338, 360
 доставка сообщений, 203, 204, 206
 коды пользователей, 234
 компоненты, 7
 многоуровневые системы, 226
 мобильные рабочие группы, 10, 12
 о Руководстве, 4
 обзор проектирования, 77, 79, 123
 обновление версий в базах данных, 221
 обновление для Adaptive Server Enterprise, 22
 описание, 4
 отслеживание сообщений, 203, 204, 206
 параметры, 271
 подписки, 9
 подписчики, 10
 предоставление полномочий publish (учебный раздел), 30, 36
 предоставление полномочий remote (учебный раздел), 30, 36
 примеры конфигурации, 12
 примеры установленных систем, 12
 принципы, 7
 процедуры резервного копирования, 214, 218, 221, 236
 публикации, 9
 репликация, 12
 системные таблицы, 281
 создание публикаций, 37
 создание публикаций (учебный раздел), 31
 статьи, 281
 типы сообщений для Windows CE, 184
 управление журналом транзакций, 214, 218, 221, 236
 установка и настройка (учебный раздел), 50

- установка и настройка для Adaptive Server Enterprise, 19
 - установка и настройка для Adaptive Server Enterprise, 18
 - установка и настройка консолидированной базы данных, 30, 36, 50
 - установка и настройка удаленной базы данных, 39
 - установка и настройка удаленной базы данных (учебный раздел), 31
 - утилита dbxtract, 265
 - утилита ssxtract, 265
 - учебный раздел для Adaptive Server Enterprise, 47
- SQL Remote Open Server
- IRIX, 393
 - архитектура, 244
 - изменения схем, 238
 - командная строка, 270
 - процедуры, 250
 - случаи использования, 243
 - установка и настройка, 248
- SQLANY.INI
- SQL Remote, 185
 - альтернатива, 186
 - установка параметров управления сообщениями, 185
- squpdate.sql
- обновление очереди с сохранением, 22
- ssqueue
- IRIX, 393
 - архитектура, 244
 - командная строка, 270
 - описание, 243, 270
 - случаи использования, 243
 - установка и настройка, 248
- ssremote
- Message Agent (учебный раздел), 57
 - безопасность, 234
 - введение, 8
 - команда, 255
 - описание, 193, 234, 255
 - учебный раздел, 57
- ssremote.sql
- установка и настройка SQL Remote, 19
- ssupdate.sql
- обновление SQL Remote для Adaptive Server Enterprise, 22
- stableq.sql
- установка и настройка SQL Remote, 20
- Sun Solaris
- совместимость, 393
- Sybase Central
- типы сообщений, 184
 - утилита извлечения, 166
- system tables
- SYSARTICLE, 281
- ## T
- TEMPDB
- требования SQL Remote, 19
- ## U
- UNIX
- поддерживаемые типы сообщений, 183
- ## V
- VERIFY_THRESHOLD
- параметр репликации, 271
- VIM
- параметры управления, 190
 - тип сообщений, 183, 190
- ## W
- Windows
- поддерживаемые SQL Remote типы сообщений, 183
 - совместимость SQL Remote, 393
- Windows CE
- ActiveSync, 393
 - репликация, 393
 - типы сообщений SQL Remote, 184
- ## A
- администрирование
- SQL Remote, 10, 210
 - SQL Remote для Adaptive Server Enterprise, 229
 - общие сведения по SQL Remote, 174
- администрирование SQL remote, 173
- администрирование:, 210
- адреса
- ftp, 186
 - SMTP, 188
 - SMTP/POP, 189
 - совместный доступ к файлам, 186
 - установка для издателя, 184

Б

базы данных

- загрузка данных в, 164
- процедуры перед извлечением, 165
- синхронизация (учебный раздел), 39

блокировка

- в системе репликации, 85, 125
- проектирование публикаций, 108, 114, 147

большие двоичные объекты

- репликация, 65

большие динарные объекты

- репликация, 271

В

внешние ключи

- перераспределение областей, 90, 129
- публикации, 84, 87, 125, 127, 129

восстановление

- SQL Remote, 194

восстановление данных

- SQL Remote, 194

время

- репликация, 65, 271

время ожидания

- Message Agent, 198

выгрузка

- консолидированные базы данных, 222

выражения подписки

- в журнале транзакций, 69
- затраты на выполнение, 69
- многозначные, 69
- описание, 81, 124
- подзапросы, 87

Г

генерация

- уникальные значения столбцов, 108

глобальное автоприращение

- использование для генерации уникальных значений, 108

группы

- извлечение, 166, 265

группы процедур

- утилита извлечения, 168

Д

дампы

- координирование, 250

даты

- репликация, 65, 271

деинсталляция

- SQL Remote для Adaptive Server Enterprise, 23
- из базы данных, 338, 360
- очередь с сохранением SQL Remote, 23

демон

- dbremote, 255
- Message Agent, 255
- ssremote, 255

добавление

- статьи, 82

Ж

журнал транзакций

- Adaptive Server Enterprise, 238
- Message Agent, 255
- SQL Remote, 10, 65, 255
- операторы обновления, 69
- отслеживание сообщений, 204
- публикации, 69
- репликация, 10, 65
- сканирование для SQL Remote Open Server, 243
- смещения, 204
- управление, 238

З

задержка при репликации

- производительность при репликации, 196

запросы на повторную передачу

- описание, 198
- сообщения, 198

запуск

- Message Agent, 210

зеркальная копия журнала транзакций

- репликация, 214, 236

значения по умолчанию

- утилита извлечения, 168

И

извлечение

- базы данных, 159, 160, 163
 - множественные базы данных, 166
 - операционные системы, 163
 - перезагрузка файлов, 164
 - пользовательские процедуры, 166
 - производительность, 131, 140
 - процедуры перед, 165
 - разработка схемы, 166
 - с использованием Sybase Central, 166
- издатель
- адрес, 184
 - добавление к базе данных (учебный раздел), 36
 - описание, 175
 - создание, 175
 - учебный раздел, 31
- изменение
- применение, 82
 - публикации, 82
 - типы сообщений, 184
- изменения схем
- SQL Remote, 238
- изменения схемы
- SQL Remote Open Server, 250
- Интернет
- SQL Remote, 188
 - электронная почта, 188
- интерфейс передачи журнала
- Message Agent, 238
- ## К
- каналы передачи сообщений
- поддерживаемые, 392
- кодирование
- описание, 201
 - пользовательское, 202
- коды пользователей
- Message Agent, 234
 - извлечение групп, 166
- командная строка
- Message Agent, 255
 - переменные среды, 255
- консолидированные базы данных
- установка и настройка (учебный раздел), 36, 50
 - учебный раздел для Adaptive Server Anywhere, 30
- конфликты
- #remote, 145
 - SQL Remote, 73
 - блокировка, 85, 125
 - методы разрешения, 103, 104, 106
 - не в выводе Message Agent, 210, 234
 - не ошибки, 210, 234
 - обнаружение в SQL Remote, 73
 - обработка в SQL Remote, 101, 144
 - параметр VERIFY_ALL_COLUMNS, 103
 - первичные ключи, 114, 147
 - первичных ключей, 108
 - предотвращение, 72
 - пример, 145
 - разрешение, 102, 103, 144
 - репликация, 72
 - уведомление, 106
 - управление, 99, 143
- конфликты UPDATE
- описание, 143
 - управление, 99
- конфликты при репликации
- управление, 99
- конфликты репликации
- описание, 143
- курсоры
- и репликация, 226
 - режим ретрансляции, 226
- кэш
- для сообщений, 197
- ## М
- мастер извлечения базы данных
- извлечение удаленной базы данных в Sybase Central, 265
 - использование, 31, 166
- мастер создания базы данных
- использование, 29
 - создание совместимой с Enterprise базы данных, 64
- мастер создания внешнего ключа
- использование, 29
- мастер создания пользователя
- использование, 31, 175
- мастер создания типа сообщений
- creating, 184
 - ftp, 186
 - MAPI, 189
 - SMTP, 188
 - VIM, 190
 - администрирование SQL Remote, 174, 175
 - изменение, 184

- использование, 184
 - описание, 183
 - параметры, 185
 - работа с, 184
 - редактирование свойств, 184
 - совместный доступ к файлам, 185, 186
 - создание, 184
 - удаление, 184, 185
- мастер создания удаленного пользователя
- использование, 31, 178
- мастеры
- извлечение базы данных, 31, 166, 265
 - создание базы данных, 29
 - создание внешнего ключа, 29
 - создание пользователя, 31, 175
 - создание публикации, 31, 79, 80, 81
 - создание статьи, 82
 - создание типа сообщений, 184
 - создание удаленного пользователя, 31, 178
- минимальная нагрузка системы
- определение, 238
- много уровневые системы
- операторы ретрансляции, 226
- многоуровневые системы
- полномочия, 180
- мобильные рабочие группы
- SQL Remote, 12
 - SQL Remote, для, 10
 - проектирование публикаций, 81, 124
- ## Н
- наборы символов
- SQL Remote, 65
 - преобразования, 65
 - совместимые, 64
- ## О
- обнаружение конфликтов
- SQL Remote, 65, 101
 - длинные типы данных, 65
 - описание, 144
- обновление
- SQL Remote, 201
 - SQL Remote для Adaptive Server Enterprise, 22
 - инсталляции SQL Remote, 160
 - параметр COMPRESSION, 201
- обновление версий
- SQL Remote, 221
 - репликация, 221
- обновления
- информация в журнале транзакций, 69
- обработка ошибок
- игнорирование ошибок, 210
 - описание, 212, 234
 - по умолчанию, 210, 234
- ограничение
- загрузка баз данных, 164
- ограничения
- Enterprise - Enterprise, 388
 - разрешение конфликтов, 145
 - утилиты извлечения, 168
- оператор COMMIT
- процедуры перехватчиков событий, 276
 - репликация, 65
- оператор CREATE SUBSCRIPTION
- описание, 118, 170
- оператор DELETE
- репликация оператора, 65
- оператор DROP PUBLICATION
- использование, 84
- оператор GRANT CONSOLIDATE, 178
- оператор GRANT PUBLISH
- консолидированная база данных (учебный раздел), 36
 - учебный раздел, 30
- оператор GRANT REMOTE
- консолидированная база данных (учебный раздел), 36
 - учебный раздел, 30
- оператор IF
- режим ретрансляции, 226
 - репликация, 226
- оператор INSERT
- репликация оператора, 65
- оператор LOOP
- режим ретрансляции, 226
- оператор PASSTHROUGH
- использование, 223
- оператор REVOKE CONSOLIDATE, 178
- оператор REVOKE PUBLISH, 175
- оператор REVOKE REMOTE, 180

- оператор ROLLBACK
 - процедуры перехватчиков событий, 276
 - оператор SYNCHRONIZE SUBSCRIPTION
 - описание, 170
 - оператор UPDATE
 - replication of, 90
 - обнаружение конфликтов, 65
 - перераспределение областей, 65
 - публикации, 90
 - репликация оператора, 65, 90, 129
 - операторы
 - репликация операторов, 65
 - операторы CREATE
 - репликация, 65
 - операторы DDL
 - SQL Remote, 238
 - репликация операторов, 65
 - операторы SQL
 - репликация, 65
 - репликация операторов, 65
 - операторы ретрансляции
 - много уровневые системы, 226
 - операторы управления
 - репликация операторов, 226
 - операционные системы
 - поддерживаемые, 391
 - совместимость, 393
 - определения репликации
 - Replication Server, 248
 - опросы
 - сообщения, 198
 - отказ носителя
 - SQL Remote, 194
 - отмена
 - полномочия CONSOLIDATE, 178
 - полномочия publish, 175
 - полномочия remote, 178
 - отмена полномочий remote, 180
 - отправка сообщений
 - учебный раздел, 42, 57
 - отслеживание сообщений
 - администрирование SQL Remote, 174, 175
 - очередь с сохранением
 - Replication Server, 245
 - SQL Remote Open Server, 245
 - ssqueue, 245
 - очистка, 255
 - системные таблицы, 298
 - установка и настройка, 20
 - ошибка удаления поврежденного сообщения, 201
 - ошибки
 - игнорирование, 210
 - не конфликты, 210, 234
 - обработка, 212, 234
 - операторы SQL и репликация, 73
 - репликация первичных ключей, 72
 - типы в репликации, 72
 - уведомление, 210, 212, 234
- ## П
- пакеты
 - режим ретрансляции, 226
 - параметр -b
 - Message Agent, 193
 - параметр BLOB_THRESHOLD
 - параметр репликации, 271
 - параметр COMPRESSION
 - параметр репликации, 271
 - параметры Message Agent, 255
 - параметр DELETE_OLD_LOGS
 - параметр репликации, 271
 - управление журналами транзакций, 218
 - параметр dsi_sql_data_style
 - конфигурирование, 250
 - параметр FIRE_TRIGGERS
 - действия триггеров, 65
 - параметр GLOBAL_DATABASE_ID
 - установка, 111
 - параметр -l
 - Message Agent, 194
 - параметр -m
 - Message Agent, 197
 - параметр OUTPUT_LOG_SEND_LIMIT в
 - удаленной БД
 - использование, 194
 - параметр OUTPUT_LOG_SEND_NOW в
 - удаленной БД
 - использование, 194

- параметр OUTPUT_LOG_SEND_ON_ERROR удаленной БД использование, 194
- параметр QUALIFY_OWNERS параметр репликации, 271
- параметр QUOTE_ALL_IDENTIFIERS параметр репликации, 271
- параметр -rd Message Agent, 198
- параметр REPLICATION_ERROR и процедуры обработки ошибок, 212 обработка ошибок, 234 отслеживание ошибок SQL, 73 параметр репликации, 271
- параметр -gr Message Agent, 198
- параметр SAVE_REMOTE_PASSWORDS параметр репликации, 271
- параметр -u Message Agent, 194
- параметр VERIFY_ALL_COLUMNS использование, 103 параметр репликации, 271
- параметр VERIFY_THRESHOLD размер сообщения, 65
- параметр переменной среды Message Agent, 255
- параметр управления debug тип сообщений file, 186 тип сообщений FTP, 187 тип сообщений MAPI, 189 тип сообщений SMTP, 188 тип сообщений VIM, 190
- параметр управления directory тип сообщений file, 186
- параметр управления Force_Download тип сообщений MAPI, 189
- параметр управления host тип сообщений FTP, 187
- параметр управления IPM_Receive тип сообщений MAPI, 189
- параметр управления IPM_Send тип сообщений MAPI, 189
- параметр управления password тип сообщений FTP, 187 тип сообщений VIM, 190
- параметр управления Path тип сообщений VIM, 190
- параметр управления pop3_host тип сообщений SMTP, 188
- параметр управления pop3_password тип сообщений SMTP, 188
- параметр управления pop3_userid тип сообщений SMTP, 188
- параметр управления port тип сообщений FTP, 187
- параметр управления receive_all тип сообщений VIM, 190
- параметр управления root тип сообщений FTP, 187
- параметр управления send_vim_mail тип сообщений VIM, 190
- параметр управления smtp_authenticate тип сообщений SMTP, 188
- параметр управления smtp_host тип сообщений SMTP, 188
- параметр управления smtp_password тип сообщений SMTP, 189
- параметр управления smtp_userid тип сообщений SMTP, 189
- параметр управления user тип сообщений FTP, 187
- параметр управления userid тип сообщений VIM, 190
- параметры SUBSCRIBE_BY_REMOTE, 98, 141 VERIFY_THRESHOLD, 65 утилита извлечения, 271
- параметры канала передачи сообщений параметр репликации EXTERNAL_REMOTE_OPTIONS, 271
- параметры репликации QUALIFY_OWNERS, 271 QUOTE_ALL_IDENTIFIERS, 271 REPLICATION_ERROR, 73, 212, 234, 271 SAVE_REMOTE_PASSWORDS, 271

- SUBSCRIBE_BY_REMOTE, 271
 VERIFY_ALL_COLUMNS, 271
 VERIFY_THRESHOLD, 271
 предоставление, 234
- пароли
 сохранение, 271
 утилита извлечения, 168
- первичные ключи
 SQL Remote, 106, 147
 генерация уникальных значений, 108
 генерация уникальных значений с помощью пулов, 112
 ошибки репликации, 72
 публикации, 65, 84, 125
 репликация, 106, 108, 147
 уникальность, 147
- перезагрузка файлов
 извлечение базы данных, 164
- переменные среды
 SQLREMOTE, 185
- перераспределение областей
 внешние ключи, 90, 129
 описание, 131
 репликация операторов UPDATE, 65
 связи, 96
 столбцы списка подписки, 129
 триггеры для, 96
- перераспределение областей
 foreign keys, 90
- перехватчики событий
 откаты недопустимы, 276
 подтверждения недопустимы, 276
 синхронизация, 276
 хранимая процедура
 sp_hook_dbremote_apply_begin, 278
 хранимая процедура
 sp_hook_dbremote_apply_end, 278
 хранимая процедура sp_hook_dbremote_begin, 276
 хранимая процедура sp_hook_dbremote_end, 276
 хранимая процедура
 sp_hook_dbremote_message_missing, 277
 хранимая процедура
 sp_hook_dbremote_message_sent, 277
 хранимая процедура
 sp_hook_dbremote_receive_begin, 276
 хранимая процедура
 sp_hook_dbremote_receive_end, 277
 хранимая процедура
 sp_hook_dbremote_send_begin, 277
 хранимая процедура
 sp_hook_dbremote_send_end, 277
- хранимая процедура
 sp_hook_dbremote_shutdown, 276
- хранимая процедура
 sp_hook_ssrm_apply_begin, 278
- хранимая процедура sp_hook_ssrm_apply_end, 278
- хранимая процедура sp_hook_ssrm_begin, 276
- хранимая процедура sp_hook_ssrm_end, 276
- хранимая процедура
 sp_hook_ssrm_message_missing, 277
- хранимая процедура
 sp_hook_ssrm_message_sent, 277
- хранимая процедура
 sp_hook_ssrm_receive_begin, 276
- хранимая процедура
 sp_hook_ssrm_receive_end, 277
- хранимая процедура sp_hook_ssrm_send_begin, 277
- хранимая процедура sp_hook_ssrm_send_end, 277
- хранимая процедура sp_hook_ssrm_shutdown, 276
- периодичность
 отправки, 178
- периодичность отправки
 Message Agent, 193
 выбор, 178
- платформы
 поддерживаемые операционные системы, 391
- поддерживаемые версии
 обновление, 160
- подзапросы
 SQL Remote, 87
 параметр SUBSCRIBE_BY_REMOTE со связями, 141
 параметр репликации, 271
 публикации, 87
 связи, 98
- подключения
 Message Agent, 193
- подписки
 Message Agent, 71
 настройка, 154
 описание, 9
 производительность, 71
 сброс, 381
 синхронизация, 164, 170
 создание, 38, 54, 118, 154
 создание (учебный раздел), 31
 управление, 118
 установка и настройка, 118
- поименованные значения по умолчанию

- утилита извлечения, 168
- поименованные ограничения
 - утилита извлечения, 168
- полномочия
 - CONSOLIDATE, 178
 - publish, 175
 - publish (учебный раздел), 30, 36
 - remote, 178
 - remote (учебный раздел), 30, 36
 - администрирование SQL Remote, 174, 175
 - многоуровневые системы, 180
 - отмена REMOTE, 180
 - предоставление CONSOLIDATE, 179
- полномочия CONSOLIDATE
 - отмена, 178
 - предоставление, 178, 179
 - управление, 175
- полномочия publish
 - отмена, 175
 - полномочия remote (учебный раздел), 36
 - полномочия remote (учебный раздел), 30
 - предоставление, 175
 - предоставление (учебный раздел), 30
- полномочия publish (учебный раздел), 36
- полномочия remote
 - отмена, 178, 180
 - предоставление, 178
- получение сообщений
 - (учебный раздел), 42
 - учебный раздел, 57
- пользователь dbo
 - системные объекты, 265
- пономочия remote
 - Sybase Central, 178
- портативные компьютеры
 - SQL Remote, 12
 - репликация, 12
- порядок сортировки
 - SQL Remote, 65
- потеряные сообщения
 - описание, 198
- предоставление
 - полномочия CONSOLIDATE, 178
 - полномочия publish, 175
 - полномочия remote, 178
- предоставления
 - полномочия publish, 175
 - представление SQL Remote articlecols
 - Adaptive Server Anywhere, 285
 - Adaptive Server Enterprise, 294
 - представление SQL Remote articles
 - Adaptive Server Anywhere, 285
 - Adaptive Server Enterprise, 294
 - представление SQL Remote publications
 - Adaptive Server Anywhere, 285
 - Adaptive Server Enterprise, 294
 - представление SQL Remote remoteoptions
 - Adaptive Server Anywhere, 286
 - Adaptive Server Enterprise, 296
 - представление SQL Remote remotetables
 - Adaptive Server Enterprise, 296
 - представление SQL Remote remotetypes
 - Adaptive Server Enterprise, 296
 - представление SQL Remote remoteusers
 - Adaptive Server Anywhere, 286
 - Adaptive Server Enterprise, 297
 - представление SQL Remote subscriptions
 - Adaptive Server Anywhere, 286
 - Adaptive Server Enterprise, 298
 - представления подписки
 - описание, 140
- Примечания
 - SQL Remote, 190
- программное обеспечение
 - dbxtract, 265
 - ssqueue, 270
 - ssxtract, 265
- программное обспечение
 - dbremote, 255
 - ssremote, 255
- проектирование
 - SQL Remote, 63
 - SQL Remote для Adaptive Server Anywhere, 77
 - SQL Remote для Adaptive Server Enterprise, 121
 - SQL Remote для Adaptive Server Enterprise
 - overview, 121
 - блокировка, 85, 125
 - в консолидированной базе данных, 64
 - конфликты и публикации, 72
 - обзор SQL Remote, 64
 - обзор для SQL Remote, 77, 79, 123
 - ошибки и публикации, 72
 - пример со связью, 96

- пример со связью, 92, 134
 производительность для SQL Remote, 87
 публикации, 84, 106, 125, 147
 связи, 92
 связь, 134
- проектирование публикации
 Adaptive Server Anywhere, 77, 79, 123
- проектирование публикаций
 SQL Remote для Adaptive Server Enterprise, 121
 для многих подписчиков, 81, 124
 с использованием выражения подписки, 81, 124
- производительность
 Message Agent, 196
 SQL Remote, 196
 ssxtract, 131, 140
 время задержки при репликации, 196
 входящие сообщения, 198
 извлечение базы данных, 166
 количество подписок, 71
 передача сообщений, 200
 потоки, 196
 при репликации, 196
 публикации, 69
 советы по проектированию для SQL Remote, 87
- профили
 Microsoft Exchange, 189
- процедура sp_add_article
 синтаксис, 331
- процедура sp_add_article_col
 синтаксис, 333
- процедура sp_add_remote_table
 синтаксис, 333
- процедура sp_create_publication
 синтаксис, 335
- процедура sp_drop_publication
 синтаксис, 336
- процедура sp_drop_remote_type
 синтаксис, 336
- процедура sp_drop_sql_remote
 деинсталляция SQL Remote для Adaptive Server
 Enterprise, 23
 синтаксис, 338
- процедура sp_grant_consolidate
 синтаксис, 339
- процедура sp_grant_remote procedure
 синтаксис, 341
- процедура sp_hook_dbxtract_begin
 использование, 111
 уникальные первичные ключи, 111
- процедура sp_link_option
 синтаксис, 342
- процедура sp_modify_article
 синтаксис, 344
- процедура sp_modify_remote_table
 синтаксис, 344
- процедура sp_passthrough
 описание, 240
 синтаксис, 346
- процедура sp_passthrough_piece
 описание, 240
 синтаксис, 348
- процедура sp_passthrough_stop
 описание, 240
 синтаксис, 348
- процедура sp_passthrough_subscription
 описание, 240
 синтаксис, 350
- процедура sp_passthrough_user
 описание, 240
 синтаксис, 351
- процедура sp_populate_sql_anywhere
 описание, 168
 синтаксис, 352
- процедура sp_publisher
 синтаксис, 352
- процедура sp_queue_clean
 синтаксис, 354
- процедура sp_queue_confirmed_delete_old
 синтаксис, 355
- процедура sp_queue_confirmed_transaction
 синтаксис, 356
- процедура sp_queue_delete_old
 синтаксис, 358
- процедура sp_queue_drop
 деинсталляция очереди с сохранением SQL
 Remote, 23
 синтаксис, 360
- процедура sp_queue_dump_database
 синтаксис, 362

- процедура `sp_queue_dump_transaction`
синтаксис, 364
- процедура `sp_queue_get_state`
синтаксис, 367
- процедура `sp_queue_log_transfer_reset`
синтаксис, 370
- процедура `sp_queue_read`
синтаксис, 373
- процедура `sp_queue_reset`
синтаксис, 373
- процедура `sp_queue_set_confirm`
синтаксис, 374
- процедура `sp_queue_set_progress`
синтаксис, 374
- процедура `sp_queue_transaction`
синтаксис, 374
- процедура `sp_remote`
синтаксис, 374
- процедура `sp_remote_option`
синтаксис, 375
- процедура `sp_remote_type`
синтаксис, 375
- процедура `sp_remove_article`
синтаксис, 376, 377
- процедура `sp_remove_remote_table`
синтаксис, 377
- процедура `sp_revoke_consolidate`
синтаксис, 378
- процедура `sp_revoke_remote`
синтаксис, 380
- процедура `sp_setreplicate`
`sp_add_remote_table`, 333
- процедура `sp_subscription`
использование, 170
описание, 154
синтаксис, 381
- процедура `sp_subscription_reset`
синтаксис, 381
- процедура `sp_user_extraction_hook`
описание, 168
- процедуры
- SQL Remote, 65
- SQL Remote Open Server, 250
- режим ретрансляции, 226
- репликация, 65
- соответствующие операторы, 388
- процедуры SQL Remote
- `sp_add_article`, 331
 - `sp_add_article_col`, 333
 - `sp_add_remote_table`, 333
 - `sp_create_publication`, 335
 - `sp_drop_publication`, 336
 - `sp_drop_remote_type`, 336
 - `sp_drop_sql_remote`, 338
 - `sp_grant_consolidate`, 339
 - `sp_grant_remote`, 341
 - `sp_link_option`, 342
 - `sp_modify_article`, 344
 - `sp_modify_remote_table`, 344
 - `sp_passthrough`, 346
 - `sp_passthrough_piece`, 348
 - `sp_passthrough_stop`, 348
 - `sp_passthrough_subscription`, 350
 - `sp_passthrough_user`, 351
 - `sp_populate_sql_anywhere`, 352
 - `sp_publisher`, 352
 - `sp_queue_clean`, 354
 - `sp_queue_confirmed_delete_old`, 355
 - `sp_queue_confirmed_transaction`, 356
 - `sp_queue_delete_old`, 358
 - `sp_queue_drop`, 360
 - `sp_queue_dump_database`, 362
 - `sp_queue_dump_transaction`, 364
 - `sp_queue_get_state`, 367
 - `sp_queue_log_transfer_reset`, 370
 - `sp_queue_read`, 373
 - `sp_queue_reset`, 373
 - `sp_queue_set_confirm`, 374
 - `sp_queue_set_progress`, 374
 - `sp_queue_transaction`, 374
 - `sp_remote`, 374
 - `sp_remote_option`, 375
 - `sp_remote_type`, 375
 - `sp_remove_article`, 376
 - `sp_remove_article_col`, 377
 - `sp_remove_remote_table`, 377
 - `sp_revoke_consolidate`, 378
 - `sp_revoke_remote`, 380
 - `sp_subscription`, 381
 - `sp_subscription_reset`, 381
- публикации
- блокировка, 85, 125
 - внешние ключи, 84, 87, 125, 127
 - изменение, 82
 - использование раздела WHERE, 80, 124
 - описание, 9
 - первичные ключи, 84, 106, 108, 114, 125, 147
 - подзапросы, 87
 - пример, 44

- пример со связью, 92, 96, 134
 - примечания, 84
 - проектирование, 64
 - проектирование, 63, 72, 84, 106, 125, 147
 - проектирование в консолидированной базе данных, 77, 79, 123
 - производительность, 69, 87
 - разделение по столбцам, 79
 - разделение по строкам, 80, 123
 - связи, 92
 - связь, 134
 - создание, 37
 - создание (учебный раздел), 31
 - ссылочная целостность, 106, 147
 - таблицы во множестве публикаций, 69
 - транзакции, 85, 125
 - удаление, 84
 - управление подписками, 118
- публикация
- выбранных столбцов, 79
- пулы первичных ключей
- генерация уникальных значений с использованием глобального автоприращения по умолчанию, 108
 - заключительная информация, 117, 151
 - описание, 147
 - пополнение, 114, 151
 - репликация, 114, 147
- ## Р
- рабочие потоки
- Message Agent, 196
- развертывание
- базы данных SQL Remote, 159, 160
- раздел WHERE
- в публикациях, 80, 124
- разделение
- Adaptive Server Enterprise, 240
 - для Adaptive Server Anywhere, 223
 - операции не реплицируются, 226
 - пакеты, 226
 - по столбцам, 79
 - по строкам, 80, 123
 - случаи применения, 225
- различия
- версии SQL Remote, 388
- разрешение конфликтов
- #remote, 145
 - методы, 103, 104, 106
 - ограничения, 145
 - пример, 145
 - реализация, 102, 144
- триггеры, 102, 103, 144
- режим непрерывной передачи
- Message Agent, 193
- режим пакетной передачи
- Message Agent, 193
- резервные копии
- SQL Remote, 194
 - для репликации, 214, 218, 236
 - для удаленных баз данных, 221
- репликация
- blob-объекты, 65
 - dbremote, 255
 - Message Agent, 255
 - ssqueue, 270
 - ssremote, 255
 - администрирование, 10
 - журнал транзакций, 65
 - журнал транзакций и, 10
 - конфликты, 72
 - мобильные рабочие группы, 12
 - обновление версий в базах данных, 221
 - общие сведения по созданию схем, 84
 - операторов SQL, 223
 - операторов курсора, 226
 - операторов управления, 226
 - операторы определения данных, 65
 - операций курсора, 226
 - ошибки первичного ключа, 106
 - ошибки первичных ключей, 147
 - ошибки ссылочной целостности, 147
 - ошибки ссылочной целостности, 106
 - параметры, 272
 - первичные ключи, 108, 114, 147
 - подписки, 9
 - примеры конфигурации, 12
 - примеры установленных систем, 12
 - процедуры, 65
 - процедуры резервного копирования, 214, 218, 221, 236
 - публикации, 9
 - режим ретрансляции, 223
 - репликация, 12
 - синхронизация, 265
 - создание схем, 84, 125
 - типы данных, 65
 - триггеры, 65, 107
 - управление журналом транзакций, 214, 218, 221, 236
 - хранимых процедур, 226
- роли
- утилита извлечения, 168
- ## С
- сброс

- подписки, 381
- типы сообщений, 184
- свойства
 - свойства типа сообщений, 184
 - создание публикаций SQL Remote, 31
 - создание публикаций SQL Remote с разделом WHERE, 80
 - создание статьи с использованием выражения прописки, 81
- связи, 92, 96, 98
- связи ‘многие ко многим’
 - пример, 92
- связь, 134, 141
- синтаксис SQL
 - храняемая процедура
 - sp_hook_dbremote_apply_begin, 278
 - храняемая процедура
 - sp_hook_dbremote_apply_end, 278
 - храняемая процедура sp_hook_dbremote_begin, 276
 - храняемая процедура sp_hook_dbremote_end, 276
 - храняемая процедура
 - sp_hook_dbremote_message_missing, 277
 - храняемая процедура
 - sp_hook_dbremote_message_sent, 277
 - храняемая процедура
 - sp_hook_dbremote_receive_begin, 276
 - храняемая процедура
 - sp_hook_dbremote_receive_end, 277
 - храняемая процедура
 - sp_hook_dbremote_send_begin, 277
 - храняемая процедура
 - sp_hook_dbremote_send_end, 277
 - храняемая процедура
 - sp_hook_dbremote_shutdown, 276
 - храняемая процедура
 - sp_hook_ssrmr_apply_begin, 278
 - храняемая процедура sp_hook_ssrmr_apply_end, 278
 - храняемая процедура sp_hook_ssrmr_begin, 276
 - храняемая процедура sp_hook_ssrmr_end, 276
 - храняемая процедура
 - sp_hook_ssrmr_message_missing, 277
 - храняемая процедура
 - sp_hook_ssrmr_message_sent, 277
 - храняемая процедура
 - sp_hook_ssrmr_receive_begin, 276
 - храняемая процедура
 - sp_hook_ssrmr_receive_end, 277
 - храняемая процедура sp_hook_ssrmr_send_begin, 277
 - храняемая процедура sp_hook_ssrmr_send_end, 277
- храняемая процедура sp_hook_ssrmr_shutdown, 276
- синхронизация
 - базы данных, 159, 160
 - использование dbxtract, 164
 - использование ssztract, 164
 - использование системы передачи сообщений для БД SQL Remote, 170
 - использование утилиты извлечения, 164
 - настройка, 276
 - операционные системы, 163
 - описание, 163
 - перехватчики событий, 276
- системные объекты
 - пользователь dbo, 265
- системные объекты для Adaptive Server Anywhere, 279
- системные объекты для Adaptive Server Enterprise, 289
- системные представления SQL Remote
 - articlecols, 285, 294
 - articles, 285, 294
 - publications, 285, 295
 - remotoptions, 286, 296
 - remotetables, 296
 - remotetypes, 296
 - remoteusers, 286, 297
 - subscriptions, 286, 298
- системные таблицы
 - SYSARTICLE, 291
 - очередь с сохранением, 298
- системные таблицы SQL Remote
 - #remote, 290
 - article, 281, 290
 - articlecol, 281, 291
 - marker, 291
 - object, 292
 - option, 292
 - passthrough, 292
 - publication, 281, 293
 - publisher, 293
 - remoteoption, 282
 - remoteoptiontype, 282
 - remotetable, 293
 - remotetype, 282, 293
 - remoteuser, 282, 294
 - sr_article, 291
 - sr_remoteoption, 293
 - sr_remoteoptiontype, 293
 - subscription, 282, 294
- системы передачи сообщений
 - поддерживаемые, 392

- службы
 - Message Agent as, 210
 - смещения
 - в журнале транзакций, 204
 - совместимость
 - Adaptive Server Enterprise и Adaptive Server Anywhere, 168
 - баз данных, 64
 - совместный доступ к файлам
 - параметры управления, 186
 - тип сообщений, 183, 185
 - создание
 - подписки, 38, 54, 118, 154
 - подписки (учебный раздел), 31
 - публикации, 37, 79
 - публикации (учебный раздел), 31
 - публикации с разделением по строкам, 80, 123
 - публикации с целыми таблицами, 79
 - статьи, 79, 82
 - статьи с разделением по строкам, 123
 - типы сообщений, 184
 - сообщения
 - в SQL Remote, 203, 204, 206
 - доставка, 203, 204, 206
 - кодирование, 201
 - контроль размера, 65
 - кэширование, 197
 - отправка (учебный раздел), 42, 57
 - отслеживание, 203, 204, 206
 - повторная передача, 198
 - получение (учебный раздел), 42, 57
 - пользовательское кодирование, 202
 - сжатие, 201, 271
 - синхронизация баз данных, 170
 - ссылочная целостность
 - SQL Remote, 106, 147, 185
 - репликация, 106, 147
 - статьи
 - properties, 79
 - действительные, 85, 127
 - добавление, 82
 - примечания относительно, 124
 - разделение по столбцам, 123
 - разделение по строкам, 123
 - системная таблица для, 281, 291
 - создание, 79
 - целая таблица, 123
 - столбцы
 - публикация выбранных столбцов, 79
 - столбцы метки времени
 - утилиты извлечения, 168
 - столбцы списка подписки
 - в SQL Remote, 127
 - описание, 129
 - поддержка, 136, 138
 - поддержка, 131
 - триггеры, 136
 - триггеры для, 131
- ## T
- таблица
 - разделение по строкам, 80
 - таблица remotetable
 - описание, 293
 - таблица remoteuser, 204
 - таблица SQL Remote article
 - Adaptive Server Anywhere, 281
 - Adaptive Server Enterprise, 290
 - таблица SQL Remote articlecol
 - Adaptive Server Enterprise, 291
 - определение, 281
 - таблица SQL Remote marker
 - Adaptive Server Enterprise, 291
 - таблица SQL Remote object
 - Adaptive Server Enterprise, 292
 - таблица SQL Remote option
 - Adaptive Server Enterprise, 292
 - таблица SQL Remote passthrough
 - Adaptive Server Enterprise, 292
 - таблица SQL Remote publication
 - Adaptive Server Anywhere, 281
 - Adaptive Server Enterprise, 293
 - таблица SQL Remote publisher
 - Adaptive Server Enterprise, 293
 - таблица SQL Remote remoteoption
 - Adaptive Server Anywhere, 282
 - таблица SQL Remote remoteoptiontype
 - Adaptive Server Anywhere, 282
 - таблица SQL Remote remotetype
 - Adaptive Server Anywhere, 282
 - описание, 293
 - таблица SQL Remote remoteuser
 - Adaptive Server Anywhere, 282
 - описание, 294

- таблица SQL Remote sr_publisher
описание, 293
- таблица SQL Remote sr_remoteoption
описание, 293
- таблица SQL Remote sr_remoteoptiontype
Adaptive Server Enterprise, 293
- таблица SQL Remote subscription
описание, 294
определение в Adaptive Server Anywhere, 283
- таблица sr_article
описание, 290
- таблица sr_articlecol
описание, 291
- таблица sr_confirmed_transaction
описание, 301
- таблица sr_marker
описание, 291
- таблица sr_object
описание, 292
- таблица sr_option
описание, 292
- таблица sr_passthrough
описание, 292
- таблица sr_publication
описание, 293
- таблица sr_queue_coordinate
описание, 301
- таблица sr_queue_state
описание, 298
- таблица sr_remotetable
описание, 293
- таблица sr_remotetype
описание, 293
- таблица sr_remoteuser
описание, 294
- таблица sr_subscription
описание, 294
- таблица sr_transaction
описание, 299
- таблица SYSREMOTEUSER, 204
- таблицы
разделение по столбцам, 79
разделение по строкам, 123
- тестирование
SQL Remote, 160
- тип данных IMAGE
репликация, 65
- тип данных LONG BINARY
репликация, 65
- тип данных LONG VARCHAR
репликация, 65
- тип данных NCHAR
утилита извлечения, 168
- тип данных NVARCHAR
утилита извлечения, 168
- тип данных TEXT
Message Agent, 196
репликация, 65
- типы данных
SQL Remote, 265
репликация, 65
- точка усечения
в журнале транзакций, 238
установка, 238
- транзакции
репликация, 65
- триггеры
RESOLVE UPDATE, 102, 103
SQL Remote, 65, 107
параметр репликации, 65
перераспределение областей, 90, 91
разрешение конфликтов, 103
разрешение конфликтов, 102, 144
репликация, 65, 91, 107
столбцы списка подписки, 136
столбцы списка подписки, 131
- триггеры RESOLVE UPDATE, 103
- ## У
- уведомление
конфликты, 106
ошибки, 234
- уведомление об ошибках, 234
- удаление

- публикации, 84
типы сообщений, 184, 185
- удаленные базы данных
полномочия remote, 178
установка и настройка (учебный раздел), 31, 39, 40
- уникальные значения
генерация с использованием глобального автоприращения, 108
генерация с помощью пулов, 112
- уникальные значения столбца
генерация, 108
- уникальные первичные ключи, 111
- управление журналом
SQL Remote, 194
- установка и настройка
SQL Remote для Adaptive Server Enterprise, 17, 19
TEMPDB для SQL Remote, 19
консолидированная база данных (учебный раздел), 30
консолидированные базы данных (учебный раздел), 36, 50
обзор SQL Remote для Adaptive Server Enterprise, 18
очередь с сохранением SQL Remote для Adaptive Server Enterprise, 20
подписки, 118
удаленные базы данных (учебный раздел), 31, 39
- устранение неисправностей
ошибки SQL Remote, 194
- утилита dbunload
репликация, 222
- утилита dbxtract
введение, 39
использование, 164
описание, 163, 265
процедура sp_hook_dbxtract_begin, 111
- утилита ssxtract
использование, 164, 168
описание, 163, 265
- утилита извлечения
группы, 166
для Adaptive Server Enterprise, 168
назначение, 166
ограничения при использовании, 166
описание, 163, 168
параметры, 271
- процедуры перед использованием, 165
- учебные разделы
SQL Remote для Adaptive Server Anywhere, 25
SQL Remote для Adaptive Server Enterprise, 47
- ## X
- храняемая процедура
sp_hook_dbremote_apply_begin
синтаксис SQL, 278
- храняемая процедура sp_hook_dbremote_apply_end
синтаксис SQL, 278
- храняемая процедура sp_hook_dbremote_begin
синтаксис SQL, 276
- храняемая процедура sp_hook_dbremote_end
синтаксис SQL, 276
- храняемая процедура
sp_hook_dbremote_message_missing
синтаксис SQL, 277
- храняемая процедура
sp_hook_dbremote_message_sent
синтаксис SQL, 277
- храняемая процедура
sp_hook_dbremote_receive_begin
синтаксис SQL, 276
- храняемая процедура
sp_hook_dbremote_receive_end
синтаксис SQL, 277
- храняемая процедура
sp_hook_dbremote_send_begin
синтаксис SQL, 277
- храняемая процедура sp_hook_dbremote_send_end
синтаксис SQL, 277
- храняемая процедура sp_hook_dbremote_shutdown
синтаксис SQL, 276
- храняемая процедура sp_hook_ssrmmt_apply_begin
синтаксис SQL, 278
- храняемая процедура sp_hook_ssrmmt_apply_end
синтаксис SQL, 278
- храняемая процедура sp_hook_ssrmmt_begin
синтаксис SQL, 276
- храняемая процедура
sp_hook_ssrmmt_message_missing
синтаксис SQL, 277

храняемая процедура `sp_hook_ssrm_message_sent`
синтаксис SQL, 277

храняемая процедура `sp_hook_ssrm_receive_begin`
синтаксис SQL, 276

храняемая процедура `sp_hook_ssrm_receive_end`
синтаксис SQL, 277

храняемая процедура `sp_hook_ssrm_send_begin`
синтаксис SQL, 277

храняемая процедура `sp_hook_ssrm_send_end`
синтаксис SQL, 277

храняемая процедура `sp_hook_ssrm_shutdown`
синтаксис SQL, 276

храняемые процедуры
SQL Remote Open Server, 250
режим ретрансляции, 226
репликация процедур, 65

Ш

шифрование
сообщений, 194

Э

электронная почта
MAPI, 183
SMTP, 183
VIM, 183