SYBASE®
An SAP® Company

管理指南第二卷

Replication Server® 15.7.1

文档 ID： DC31035-01-1571-01
最后修订日期： 2012 年 4 月

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

目录

约定 1
验证和监控 Replication Server 3
检查复制系统日志文件中的错误 3
验证复制系统 4
监控 Replication Server 5
验证服务器状态 5
可视化状态监控 6
显示复制系统线程状态 6
设置和使用阈值级别 9
监控分区百分比 9
自定义数据库操作 11
函数、函数字符串和函数字符串类 11
使用函数、函数字符串和类 11
函数 12
系统函数摘要 14
函数字符串 18
具有多个函数字符串的系统函数 19
函数字符串类 19
系统提供的类 20
函数字符串继承 21
混合版本系统中的限制 22
管理函数字符串类 23
创建函数字符串类 23
将函数字符串类指派给数据库 26
删除函数字符串类 27
管理函数字符串 27
函数字符串和函数字符串类 27
函数字符串输入模板和输出模板 28
输出模板 28
输入模板 30
函数字符串变量 31

创建函数字符串 33
更改函数字符串 36
删除函数字符串 37
恢复缺省函数字符串 38
使用输出模板创建空函数字符串 39
在函数字符串中定义多个命令 39
非 ASE 服务器的命令批处理 40
在语言输出模板中使用声明语句 41
显示与函数相关的信息 42
使用 admin 命令获取信息 42
使用存储过程获取信息 42
缺省系统变量 43
扩展缺省函数字符串 43
使用 replicate minimal columns 子句 44
使用具有 text、unitext、image 和 rawobject 数据类型的函数字符串 44
使用 rs_writetext 函数字符串的 writetext 输出模板选项 45
使用 rs_writetext 函数字符串的 none 输出模板 . . .45
管理热备份应用程序 47
热备份应用程序 47
热备份的工作原理 47
热备份应用程序中的数据库连接 49
主数据库和复制数据库以及热备份应用程序 49
热备份要求和限制 50
用于维护备用数据库的函数字符串 52
为热备份复制的信息 52
复制方法的比较 53
使用 sp_reptostandby 启用复制 54
使用 sp_setreptable 启用复制 59
使用 sp_setrepproc 复制用户存储过程 59
复制名称相同但所有者不同的表 60
在热备份应用程序中复制 text、unitext、image 和 rawobject 数据 60

配置热备份数据库以进行 SQL 语句复制 61
复制加密列 61
复制带引号的标识符 62
热备份包含复制数据库时 62
更改当前 isql 会话的复制 62
设置 ASE 热备份数据库 63
开始之前 63
任务 1：创建逻辑连接 64
任务 2：添加活动数据库 65
任务 3：为活动数据库中的对象启用复制 65
任务 4：添加备用数据库 66
在 ASE 热备份环境中复制主数据库 74
在热备份环境中设置主数据库复制 75
切换活动数据库和备用 ASE 数据库 76
确定是否需要切换 76
切换活动数据库和备用数据库之前 76
内部切换步骤 77
切换活动数据库和备用数据库之后 78
进行切换 78
监控热备份应用程序 82
Replication Server 日志文件 82
监控热备份应用程序的命令 83
设置客户端以使用活动数据服务器 84
两个 interfaces 文件 84
客户端应用程序的数据服务器符号名 84
将客户端数据服务器映射到当前的活动数据服务器 85
更改热备份数据库连接 85
更改逻辑连接 85
更改物理连接 88
删除逻辑数据库连接 89
使用复制的热备份应用程序 90
主数据库的热备份应用程序 90
复制数据库的热备份应用程序 92

热备份数据库的复制定义和预订 95
热备份的 alter table 支持 95
使用复制定义优化性能 97
使用复制定义来复制冗余更新 99
在热备份应用程序中使用预订 99
创建备用数据库时丢失列 102
丢失检测和恢复 103
性能调优 105
Replication Server 内部处理 105
线程、模块和守护程序 105
主 Replication Server 中的进程 106
复制 Replication Server 中的处理 110
影响性能的配置参数 111
影响性能的 Replication Server 参数 111
影响性能的连接参数 122
影响性能的路由参数 130
使用调优参数的建议 131
设置 SQM Writer 等待的时间 131
高速缓存系统表 131
执行程序命令高速缓存 133
稳定队列高速缓存 134
异步分析程序、ASCII 打包和直接命令复制 135
设置唤醒时间间隔 141
设置 SQT 高速缓存大小 142
控制未完成的字节数 142
控制网络操作数 143
控制 RepAgent 执行程序可以处理的命令数 143
指定分配的稳定队列段数 144
为稳定队列选择磁盘分区 144
使 SMP 更有效 144
指定组中的事务数 145
设置事务大小 146
启用非阻塞提交 146
内存消耗量控制 146

并行 DSI 线程 148
使用并行 DSI 线程的优点和风险 149
并行 DSI 参数 149
并行 DSI 的组件 153
使用并行 DSI 线程处理事务 154
选择隔离级别 154
事务序列化方法 156
分区规则：减少争用和增大并行度 159
解决更新冲突 163
配置并行 DSI 以优化性能 167
Parallel DSI and the rs_origin_commit_time 系统变量 170
DSI 批量拷入 171
DSI 批量拷入配置参数 171
对预订实现的更改 172
用于批量拷入的计数器 172
批量拷入限制 173
SQL 语句复制 174
SQL 语句复制简介 174
基于日志的复制的性能问题 174
启用 SQL 语句复制 177
设置 SQL 语句复制阈值 180
为 SQL 语句复制配置复制定义 184
SQL 语句复制的行计数验证 186
SQL 语句复制作用域 187
SQL 语句复制解决的问题 189
使用 SQL 语句复制的例外情况 190
RSSD 系统表修改 192
用于 SQL 语句复制的 Adaptive Server 监控表 193
产品版本和混合版本要求 193
降级和 SQL 语句复制 193
适用于增强的 Replication Server 性能的动态 SQL 194
动态 SQL 配置参数 194
设置配置参数以使用动态 SQL 194

表级动态 SQL 控制 195
replicate minimal columns 子句和动态 SQL 195
动态 SQL 的限制 196
高级服务选件 196
向 Adaptive Server 进行高容量自适应复制 197
提高的 DSI 效率 210
提高的 RepAgent 执行程序线程效率 211
提高的分配器线程读取效率 212
增强的内存分配 212
增加队列块大小 212
Multi-Path Replication 217
Multi-Path Replication 快速入门 218
并行事务流 220
缺省和替代连接 222
到复制数据库的多个连接 223
来自主数据库的多个连接 226
复制定义和预订 227
多个主复制路径 229
为 MSA 环境创建多个复制路径 241
用于热备份环境的多个复制路径 241
专用路由 243
用于多个复制路径的 Adaptive Server 监控表 245
替代主连接和复制连接的系统表支持 246
多处理器平台 246
启用多处理器支持 247
用于监控线程状态的命令 247
监控性能 247
分配队列段 248
缺省分配机制 248
选择磁盘分配 248
删除提示和分区 250
RMS 中的心跳功能 250
使用计数器监控性能 251
用于查看计数器值的命令 251

模块 251
Replication Server 模块 252
计数器 253
数据采样 253
收集特定时间段内的统计信息 254
收集不定时间段内的统计信息 256
在屏幕上查看统计信息 258
查看吞吐量速率 258
查看有关消息和内存使用情况的统计信息 259
查看稳定队列中的事务数 259
查看在 RSSD 中保存的统计信息 259
使用 rs_dump_stats 过程 260
查看有关计数器的信息 261
重置计数器 262
生成性能报告 262
错误和例外处理 263
常规错误处理 263
错误日志文件 263
Replication Server 错误日志 264
RepAgent 错误日志消息 266
数据服务器错误处理 267
用于错误处理的 RCL 命令和系统过程 267
缺省错误类 268
非 ASE 数据库的本机错误代码 268
创建错误类 268
更改错误类 270
初始化新的错误类 270
删除错误类 270
更改错误类的主 Replication Server 271
显示错误类信息 271
为数据服务器错误指定操作 272
显示为错误号指定的操作 273
行计数验证 274
例外处理 276

处理失败的事务 277
访问例外日志 278
从例外日志中删除事务 280
DSI 重复检测 281
系统事务的重复检测 281
复制系统恢复 283
如何使用恢复过程 283
配置复制系统以支持 Sybase 故障切换 284
为 Replication Server 启用故障切换支持 284
配置复制系统以防止数据丢失 286
恢复的保存间隔 286
备份 RSSD 289
创建协调的转储 290
从分区丢失或分区故障中恢复 290
分区丢失或故障的症状和相关恢复过程 291
从分区丢失或分区故障中恢复 291
从脱机数据库日志中恢复消息 292
从联机数据库日志中恢复消息 294
从截断的主数据库日志中恢复 294
从截断的主数据库日志中恢复消息 295
从主数据库故障中恢复 296
从转储装载主数据库 296
从协调的转储装载 297
从 RSSD 故障中恢复 298
从转储中恢复 RSSD 的过程 299
使用基本 RSSD 恢复过程 300
使用预订比较过程 302
使用预订重新创建过程 308
使用拆散和重聚过程 310
恢复支持任务 310
重建稳定队列 311
Adaptive Server 的复制数据库重新同步 321
配置数据库重新同步 321
数据库重新同步方案 324

异步过程 331
异步过程传递简介 331
Adaptive Server 记录的复制存储过程 331
应用存储过程 332
请求存储过程 332
异步存储过程的先决条件 333
实现应用存储过程 334
警告情况 336
实现请求存储过程 337
将存储过程和表指定为要复制 339
管理用户定义的函数 339
创建用户定义的函数 339
将参数添加到用户定义的函数中 340
删除用户定义的函数 341
将函数映射到不同的存储过程名称 342
为用户定义的函数指定非唯一名称 343
Sun Cluster 2.2 的高可用性 345
Sybase Replication for Sun Cluster HA 简介 345
术语 345
技术概述 346
配置 Replication Server 以获得高可用性 347
安装 Replication Server 以获得 HA 347
安装作为数据服务的 Replication Server 348
将 Replication Server 作为数据服务进行管理 350
启动和关闭数据服务 350
Sun Cluster for HA 日志 351
实现参考复制环境 353
参考复制环境实现 353
平台支持 353
参考实现组件 353
参考环境的前提条件 354
构建参考环境 354
参考实现配置文件 355
配置参考环境 358

在参考环境中运行性能测试 359
从参考环境中获取测试结果 359
rs_ticket_history 报告 359
监控器和计数器报告 360
关闭参考实现服务器 361
清除参考环境 361
为参考环境创建的对象 362
表模式 363
词汇表 369
获取帮助及其它信息 381
技术支持部门 381
下载 Sybase EBF 和维护报告 381
Sybase 产品和组件认证 382
创建 MySybase 配置文件 382
辅助功能特性 382
索引 383

约定

Sybase® 文档中使用以下样式和语法约定。

样式约定

关键字	定义
等宽字体（固定宽度）	• SQL 和程序代码 • 完全按照所示输入的命令 • 文件名 • 目录名
等宽斜体	在 SQL 或程序代码段中，用户指定的值的占位符（请参见下面的示例）。
斜体	• 文件名和变量名 • 对其它主题或文档的交叉引用 • 在文本中，用户指定的值的占位符（请参见下面的示例） • 文本中的词汇表术语
粗体 san serif	• 命令、函数、存储过程、实用程序、类和方法的名称 • 词汇表条目（在词汇表中） • 菜单选项路径 • 在编号任务或过程步骤中，您单击的用户界面 (UI) 元素，如按钮、复选框、图标等

如有必要，接下来会在文本中对占位符（特定于系统或设置的值）进行说明。例如：运行：

```
installation directory\start.bat
```

其中 *installation directory* 是应用程序的安装位置。

语法约定

关键字	定义
{ }	大括号表示必须至少选择括号中的一个选项。不要在输入命令时键入大括号。
[ ]	中括号表示可以选择括号中的一个或多个选项，也可不选。不要在输入命令时键入中括号。

关键字	定义
( )	小括号应作为命令的一部分输入。
|	竖线表示只能选择一个显示的选项。
,	逗号表示可以选择任意多个显示的选项，逗号作为命令的一部分输入以分隔选项。
...	省略号（三点）表示可以将最后一个单元重复任意多次。不要在命令中包括省略号。

区分大小写

- 所有命令语法和命令示例都以小写形式显示。但是，复制命令名称不区分大小写。例如，**RA_CONFIG**、**Ra_Config** 和 **ra_config** 是等效的。
- 配置参数的名称区分大小写。例如，**Scan_Sleep_Max** 与 **scan_sleep_max** 不同，前者将被解释为无效参数名称。
- 复制命令中的数据库对象名称不区分大小写。但是，若要在复制命令中使用混合大小写的对象名（以与主数据库中混合大小写的对象名相匹配），请用引号字符分隔该对象名。例如： **pdb_get_tables** "*TableName*"
- 根据有效的排序顺序，标识符和字符数据可能要区分大小写。
  - 如果使用区分大小写的排序顺序（如“binary”），则必须用正确的大写和小写字母组合形式输入标识符和字符数据。
  - 如果使用不区分大小写的排序顺序（如“nocase”），则可以用任意大写或小写字母组合形式输入标识符或字符数据。

术语

Replication Agent™ 是用于描述 Replication Agent for Adaptive Server® Enterprise、Replication Agent for Oracle、Replication Agent for IBM DB2 UDB 和 Replication Agent for Microsoft SQL Server 的通用术语。特定名称包括:

- RepAgent – 用于 Adaptive Server Enterprise 的 Replication Agent 线程
- Replication Agent for Oracle
- Replication Agent for Microsoft SQL Server
- Replication Agent for UDB – 用于 Linux、Unix 和 Windows 上的 IBM DB2

验证和监控 **Replication Server**

Replication Server® 验证和监控包括检查错误日志，验证复制系统的组件是否正在运行，以及监控系统组件和进程的状态。

复制系统包括数据服务器和 Replication Server，可能还包括用于异构数据服务器的 Replication Agent。Adaptive Server 的 Replication Agent 是 RepAgent，它是一个 Adaptive Server 线程。

注意： 如果您使用的是异构数据服务器的 Replication Agent，请参见适用于您的数据服务器的 Replication Agent 文档，以获得有关排除 Replication Agent 故障的信息。

在一个完全正常运行的复制系统中，所有数据服务器、Replication Server、Replication Agent 和它们的内部线程及其它组件都应处于运行状态。可以在复制系统上执行的基本故障排除任务包括：

- 检查错误日志中的状态消息和错误消息
- 登录到系统服务器，并检查所有线程是否正常运行，路由和连接是否已就绪以及 interfaces 文件信息是否正确
- 监控 Replication Server 及其线程并检查分区阈值级别

有关监控 Replication Server 和排除 Replication Server 故障的详细信息，请参见《Replication Server 故障排除指南》。

检查复制系统日志文件中的错误

Replication Server 在 Replication Server 错误日志文件中记录状态消息和错误消息，包括内部错误。

使用 **admin log_name** 命令可显示当前日志文件的路径。该日志文件缺省名为 `repserver.log`。您可以更改此缺省名称，方法是附带 **-E** 选项执行 **repserver**，并指定新日志文件名。

有关这些命令的详细信息，请参见《Replication Server 参考手册》中的“Replication Server 命令”。

出现内部错误时，唯一可对 Replication Server 执行的操作就是转储堆栈并退出运行。为了便于诊断，Replication Server 会在日志中输出它的执行堆栈的轨迹，并保留它在发生错误时的状态记录。

在您删除消息之前，它们会不断在错误日志文件中累积。因此，您可以选择在关闭 Replication Server 时截断日志文件。您还可使用 **admin set_log_name** 命令关闭 Replication Server 日志文件并开始启用一个新日志文件。

Replication Server 日志文件包含异步命令（例如 **create subscription** 和 **create route**）执行过程中生成的消息，这些消息在这类命令完成之后继续处理。在您执行异步命令时，请特别注意受此过程影响的 Replication Server 日志文件。

如果没有日志文件，重要的错误信息会被写入标准错误输出文件，您可以在某个终端上显示该文件或是将其重定向到某个文件。

验证复制系统

在创建复制定义或预订或者对系统进行诊断时，请验证整个复制系统是否正常工作。

前提条件

在确认复制系统是否正常工作之前，请确保没有线程处于关闭状态。

过程

如果您遇到错误，可以通过验证系统来排除线程或组件没有运行的可能性，或者没有正确设置路由和连接的可能性。

要检查 Replication Server 线程是否正在运行，您可以执行 **admin who_is_down** 以仅显示未运行的线程。执行 **admin who** 则会显示所有线程的信息。

1. 确认复制系统服务器和 Replication Agent 正在运行且可以使用。

   在主节点上，登录到下列服务器：
   - 具有主数据库及其 Replication Agent 的数据服务器
     如果您使用的是 Adaptive Server，请在 Adaptive Server 中执行 **sp_help_rep_agent** 以显示 RepAgent 线程的状态信息。
   - 管理主数据库的 Replication Server
   - 主 Replication Server 的 RSSD（及其 Replication Agent）
     如果您使用的是 Adaptive Server，请在 Adaptive Server 中执行 **sp_help_rep_agent** 以显示 RepAgent 线程的状态信息。

   在复制节点上，登录到下列服务器：
   - 具有复制数据库的数据服务器（如果在这些数据库中执行了请求函数，则还包含这些数据库的 Replication Agent）
     如果您使用的是 Adaptive Server，请在 Adaptive Server 中执行 **sp_help_rep_agent** 以显示 RepAgent 线程的状态信息。
   - 管理复制数据库的 Replication Server
   - 复制 Replication Server 的 RSSD（及其 Replication Agent）
     如果您使用的是 Adaptive Server，请在 Adaptive Server 中执行 **sp_help_rep_agent** 以显示 RepAgent 线程的状态信息。
2. 在 Replication Server 上执行 **admin show_connections** 命令以验证这些路由和连接已经就绪：

- 从主 Replication Server 到每一个复制 Replication Server 的路由
- 主 Replication Server 和主数据库之间的数据库连接
- 从复制 Replication Server 到主 Replication Server 的路由（如果复制 Replication Server 管理执行请求函数的复制数据库）
- 每一个复制 Replication Server 和它的复制数据库之间的数据库连接

3. 验证 interfaces 文件中的条目的准确性。

   创建预订时，请确保复制 Replication Server 的 interfaces 文件中存在主数据库服务器的条目。如果您使用的是原子实现或非原子实现，则复制 Replication Server 将通过到主数据服务器的直接连接检索初始行。

4. 使用 **admin who** 验证以下 Replication Server 线程是否正在运行：
   - 数据服务器接口 (DSI)
   - Replication Server 接口 (RSI)
   - 分配器 (DIST)
   - 稳定队列管理器 (SQM)
   - 稳定队列事务 (SQT) 接口
   - RepAgent 用户

另请参见

- 显示复制系统线程状态 （第 6 页）

监控 Replication Server

在复制系统运行时，您可能需要监控它的组件和进程。

您需要执行以下操作：

- 监控复制系统服务器。
- 监控 DSI、RSI 和其它线程状态。
- 使用系统信息命令获得有关 Replication Server 的各方面的信息。

验证服务器状态

可以通过几种方法验证服务器的状态。

- 使用 **isql** 登录到每台服务器。如果成功登录，则说明服务器正在运行。
- 创建一个用于登录到各个 Adaptive Server 及其 RepAgent 线程、其它 Replication Agent（如果存在）和 Replication Server 并显示其状态的脚本。确保脚本中的所有服务器都包括在 interfaces 文件中。

如果登录失败，则可能是由以下某种原因引起的：

问题：您键入的名称不正确，或您使用的 interfaces 文件中没有相应服务器的条目。

```
DB-LIBRARY error:
  Server name not found in interface file.
```

问题：服务器正在运行，但您指定的登录名或口令不正确。

```
DB-LIBRARY error:
  Login incorrect.
```

问题：服务器未运行。

```
Operating-system error:
  Invalid argument
 DB-LIBRARY error:
  Unable to connect: Server is unavailable
  or does not exist.
```

问题：无法找到 interfaces 文件。

```
Operating-system error:
  No such file or directory
 DB-LIBRARY error:
  Could not open interface file.
```

问题：存在 interfaces 文件，但您无权访问此文件。

```
Operating-system error:
  Permission denied
 DB-LIBRARY error:
   Could not open interface file
```

如果无法登录并且没有收到错误消息，则可以推测服务器已停止处理。如果需要帮助以确定问题，请与 Sybase 技术支持联系。

可视化状态监控

使用 Replication Manager GUI 监控 Replication Monitoring Services (RMS) 中的状态。Replication Manager 通过 RMS 连接到此环境中的服务器。

Replication Manager 将以图形方式显示环境或对象状态。

环境的状态也就是其组件的状态。对象的状态包括其当前状态和出现此状态的原因列表。每个对象的状态都会显示在对象图标上、父对象“详细信息”列表中以及该对象的“属性”对话框中。您可以监控服务器、连接、路由和队列的状态。

请参见《Replication Server 管理指南第一卷》中的“使用 Sybase Central 管理 Replication Server”。

显示复制系统线程状态

使用相关的 **admin who** 命令或系统过程显示有关当前 Replication Server 线程的不同类型的一般性信息。

表 1. 监控 Replication Server 线程

Replication Server 线程	命令
分配器 (DIST) – 使用 SQT 和 SQM 从入站队列中读取事务。	**admin who, dist**

Replication Server 线程	命令
数据服务器接口 (DSI) – 将事务提交给数据服务器。	**admin who, dsi**
REP AGENT USER – 验证来自数据服务器的事务是否有效，并将它们写入到入站队列中。	**admin who** **注意：** 请使用 **sp_who** 或 **sp_help_rep_agent** 显示 Adaptive Server 上的 RepAgent 线程的状态。
Replication Server 接口 (RSI) – 登录到每台目标 Replication Server 并将命令从稳定队列传送到目标服务器。	**admin who, rsi**
稳定队列管理器 (SQM) – 管理 Replication Server 稳定队列。	**admin who, sqm**
稳定队列事务 (SQT) 接口 – 读取队列中的事务并将它们传递给 SQT 读取器。	**admin who, sqt**

请参见：

- 《Replication Server 故障排除指南》以获得对该命令输出的解释，以便进行故障排除。
- 《Replication Server 参考手册》的“Replication Server 命令”中的“**admin who**”
- 《Replication Server 参考手册》的“Adaptive Server 命令和系统过程”中的“**sp_help_rep_agent**”
- 《Adaptive Server Enterprise 参考手册：过程》的“系统过程”中的“**sp_who**”

使用系统信息命令

除了 **admin who** 外，Replication Server 还提供了其它 **admin** 命令，以帮助您监控 Replication Server。

有关每个命令的详细信息，请参见《Replication Server 参考手册》中的“Replication Server 命令”。

表 2. 系统信息命令概述

命令	说明
admin disk_space	显示 Replication Server 访问的磁盘分区的使用情况。
admin echo	确定本地 Replication Server 是否正在运行。
admin get_generation	检索在恢复操作中使用的主数据库的生成号。
admin health	显示 Replication Server 的总体状态。
admin log_name	显示当前日志文件的路径。
admin logical_status	显示热备份应用程序中使用的逻辑数据库连接的状态。

命令	说明
admin pid	显示 Replication Server 的进程 ID。
admin quiesce_check	确定 Replication Server 中的队列是否已停顿。
admin quiesce_force_rsi	确定 Replication Server 是否已停顿。还可强制 Replication Server 传递出站消息。
admin rssd_name	显示 RSSD 的数据服务器和数据库的名称。
admin security_property	显示 Replication Server 支持的基于网络的安全性系统的安全性功能。
admin security_setting	显示特定目标服务器的基于网络的安全性设置。
admin set_log_name	关闭现有的 Replication Server 日志文件并打开新的日志文件。
admin show_connections	显示与 Replication Server 之间的所有连接和路由的信息。
admin show_function_classes	显示现有函数字符串类及其父类的名称，并指明继承的级别号。
admin show_route_versions	显示起始于 Replication Server 和终止于 Replication Server 的路由的版本号。
admin show_site_version	显示 Replication Server 的节点版本。
admin sqm_readers	显示正在读取入站队列的线程的信息。
admin stats	显示有关 Replication Server 计数器的信息和统计数据。替换 **admin statistics**。
admin statistics, md	显示有关消息传递和计数器的统计信息。
admin statistics, mem	显示有关内存使用情况的统计信息。
admin statistics, reset	重新设置消息传递统计信息。
admin version	显示所运行的 Replication Server 版本，表示软件版本。
admin who	显示有关 Replication Server 中的所有线程的信息。
admin who, dsi	显示有关连接到某个数据服务器的 DSI 线程的信息。
admin who, rsi	显示有关连接到其它 Replication Server 的 RSI 线程的信息。
admin who, sqm	显示有关 SQM 管理的所有队列的信息。
admin who, sqt	显示有关 SQT 管理的所有队列的信息。
admin who_is_down	显示的信息与 **admin who** 相同，但只有关于已关闭线程的信息。
admin who_is_up	显示的信息与 **admin who** 相同，但只有关于当前正在运行的线程的信息。

设置和使用阈值级别

您可以将 Replication Server 配置为在分区过满时发出警告。

当 Replication Server 收到的信息比它发送的信息多时，稳定队列分区就会充满。例如，如果主节点和复制节点之间的网络关闭，未传递的消息就会在主节点的 Replication Server 中排队等候。当网络恢复服务之后，这些消息将被传送出去，并随后从主 Replication Server 的分区中删除。

如果分区被完全填满，发送方就不能将它们的消息传递到 Replication Server，这样消息就会开始在原先节点的分区和主数据库事务日志中备份。

警告! 如果这种情况得不到纠正，RepAgent 就无法更新数据库日志中的辅助截断点，事务日志文件就将被填满。这样，客户端就无法执行主数据库上的事务。

可以将 **configure replication server** 与 **sqm_warning_thr1**、**sqm_warning_thr2** 和 **sqm_warning_thr_ind** 一起使用，以便将 Replication Server 配置为在分区过满时发出警告。请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**configure replication server**”。

监控分区百分比

可以使用日志文件中的消息监控 Replication Server 分区百分比变化。

Replication Server 在 1MB 的分区段上操作。每当分配或释放分配分区段时，它都会计算：

- 使用中的分区段占总分区段的百分比
- 受影响的稳定队列使用的分区段占总分区段的百分比

如果使用的分区段百分比上升到 **sqm_warning_thr1** 或 **sqm_warning_thr2** 指定的百分比之上，便会在日志文件中写入一条消息：

```
WARNING: Stable Storage Use is Above threshold percent
```

如果您经常看到这条消息，您可能需要向 Replication Server 中添加分区，或者排除导致队列过满的重复故障。

当第一个百分比下降到 **sqm_warning_thr1** 或 **sqm_warning_thr2** 指定的百分比以下时，便会写入一条消息，以告知您导致原来警告的情况已不存在了：

```
WARNING CANCEL: Stable Storage Use is Below threshold percent
```

当由一个稳定队列使用的空间占总空间的百分比超出了 **sqm_warning_thr_ind** 所指定的百分比时，受影响的稳定队列所使用的分区段占总分区段的百分比将触发一条警告消息：

```
WARNING: Stable Storage Use by queue name is Above threshold percent
```

该警告提醒您存在问题，这些问题导致特定稳定队列被持续填充，直至其在总分区空间中所占份额过大为止。例如，如果某路由长时间被挂起，它的稳定队列可能会被持续填充，直至它占用了足以触发警告的分区空间为止。

当稳定队列所使用的分区空间占总空间的百分比下降到 **sqm_warning_thr_ind** 百分比以下时，Replication Server 便会在日志文件中写入一条取消消息：

```
WARNING CANCEL: Stable Storage Use by queue name is Below threshold
percent.
```

自定义数据库操作

创建和更改函数、函数字符串、函数字符串类，以允许将复制定义用于 Adaptive Server 以外的数据库服务器。

函数、函数字符串和函数字符串类

Replication Server 将主数据库中的命令转换为表示数据服务器操作（例如，插入、删除、选择、开始事务，等等）的 Replication Server 函数。它会将这些函数分配给系统中的远程 Replication Server，这样，这些函数就会在远程数据库中执行那些操作。

无论实际更新复制数据的数据服务器是什么类型，主 Replication Server 均以相同的格式分配函数。函数不是特定于数据库的，它们包括执行操作所需的所有数据，但是不指定在目标数据服务器中完成操作所需的语法。

远程 Replication Server 将函数转换为特定于执行它们的数据服务器的命令。函数字符串包含用于执行函数的数据库特定指令。管理着一个数据库的复制 Replication Server 会使用一个相应的函数字符串将该函数映射到数据服务器的一组指令。例如，**rs_insert** 函数的函数字符串提供要在复制数据库中应用的实际语言。

函数与数据服务器命令的这种分离使得您可以在异构数据服务器中维护复制的数据。使用 Replication Server，可以自定义函数字符串，指定 Replication Server 函数映射到 SQL 命令的方式。如果您需要自定义的数据服务器操作，可以创建函数字符串。通过更改在目标数据库中执行操作的方式，您可以自定义复制的数据应用程序。

函数字符串划分为函数字符串类的组，因此，您可以根据数据服务器将函数到命令的映射分组。Replication Server 为 Adaptive Server Enterprise、Oracle、Microsoft SQL Server、IBM DB2 UDB 和其它数据库提供了函数字符串类。您可以创建新派生的函数字符串类，在其中自定义某些函数字符串，另外从这些类或其它类继承所有其它函数字符串。您也可以创建全新的类，在其中创建所有的新函数字符串。

您可能还需要为复制函数创建函数字符串，利用它们，您可以在远程数据库上执行存储过程。对于 Replication Server 不在目标数据库使用的函数字符串类中自动为其生成函数字符串的任何复制函数，您必须为其创建函数字符串。

使用函数、函数字符串和类

可以通过几种方法使用函数和函数字符串以自定义数据库操作。

您可以：

- 新建用于特定类型的数据库的函数字符串类，并自定义部分或全部函数字符串。

- 对于原子实现，请使用与主数据库连接关联的函数字符串类中的函数，而不是与复制数据库连接关联的函数字符串类中的函数。
- 为系统提供的函数字符串类 **rs_sqlserver_function_class** 更改函数字符串。
- 创建一个直接或间接地从系统提供的函数字符串类 **rs_default_function_class** 继承函数字符串的函数字符串类。
- 将系统提供的函数字符串类用于非 ASE 数据服务器：**rs_iq_function_class**、**rs_db2_function_class**、**rs_mss_function_class** 或 **rs_oracle_function_class**。有关使用异构数据类型支持 (HDS) 功能转换数据类型的详细信息，请参见《Replication Server 管理指南第一卷》的“管理复制表”中的“使用 HDS 转换数据类型”。

可以通过使用 **isql** 在命令行中输入的 Sybase Central™ 或 RCL 命令来使用函数、函数字符串和类。

有关系统函数的详细信息，请参见《Replication Server 参考手册》中的“Replication Server 系统函数”。

另请参见

- 管理函数字符串类 （第 23 页）
- 管理函数字符串 （第 27 页）

函数

Replication Server 使用两种主要类型的函数。

Replication Server 的主要函数类型为：

- 系统函数
- 用户定义的函数

您可以根据需要为任一类型的函数创建自定义函数字符串。

另请参见

- 管理函数字符串 （第 27 页）

系统函数

系统函数代表这样的数据服务器操作：使用的函数字符串由 Replication Server 提供或当您在复制系统上安装新数据库时可用。

除非您的应用程序要求，否则不必为系统函数自定义函数字符串。系统提供的类会为您生成这些函数字符串。

系统函数包括：

- 代表数据处理操作（例如，插入、更新、删除、选择和使用 holdlock 选择）的函数。这些系统函数具有复制定义作用域。

- 代表事务控制指令的函数。这些函数包括开始事务和提交事务等操作。这些系统函数具有函数字符串类作用域。

另请参见

- 函数作用域 （第 13 页）
- 系统函数摘要 （第 14 页）

用户定义的函数

使用用户定义的函数，您可以使用 Replication Server 在复制系统中的节点之间分配复制的存储过程。

您必须为用户定义的函数创建函数字符串，除非您使用一个直接或间接地从 **rs_default_function_class** 继承函数字符串的函数字符串类。用户定义的函数包括：

- 在复制与函数复制定义关联的存储过程时使用的函数。在您创建函数复制定义时，Replication Server 会自动创建一个这种类型的用户定义函数。请参见《Replication Server 管理指南第一卷》中的“管理复制函数”。
- 在复制与表复制定义关联的存储过程时使用的函数。您要自己创建和维护这种类型的用户定义函数。
  可以使用异步过程复制与表复制定义关联的存储过程。

用户定义的函数将复制定义作用域作为函数作用域类型。

您为用户定义的函数创建的任何函数字符串都应该在创建复制定义的主 Replication Server 上创建。如果使用函数复制定义，请参见《Replication Server 管理指南第一卷》的“管理复制函数”中的“使用复制函数”。

另请参见

- 异步过程 （第 331 页）
- 函数作用域 （第 13 页）

函数作用域

函数的作用域定义了要应用该函数的对象：复制定义或函数字符串类。

您必须知道函数作用域才能确定在主 Replication Server 或复制 Replication Server 上自定义函数字符串的位置。

函数字符串类作用域

具有函数字符串类作用域的函数为该类定义一次。具有函数字符串类作用域的函数包括代表事务控制指令（例如，**rs_begin**、**rs_commit** 或 **rs_marker**）但不执行数据处理的系统函数。用户定义的函数的函数字符串没有类作用域。

必须在主 Replication Server 为函数字符串类自定义具有函数字符串类作用域的函数的函数字符串。

复制定义作用域
尽管具有复制定义作用域的函数可能会有多个函数字符串，但对于某个特定的表复制定义或函数复制定义，只定义一次该函数。

具有复制定义作用域的函数包括：

- 执行数据处理操作的系统函数（例如，**rs_insert**、**rs_delete**、**rs_update**、**rs_select**、**rs_select_with_lock**，以及在复制 `text`、`unitext` 和 `image` 数据时使用的特殊函数）。
- 用于表复制定义或函数复制定义的用户定义函数。
  具有复制定义作用域的系统函数必须在创建复制定义的 Replication Server 上自定义。具有复制定义作用域的用户定义函数必须在创建复制定义的 Replication Server 上自定义。

有关所有系统函数的完整文档，请参见《Replication Server 参考手册》中的“Replication Server 系统函数”。

另请参见
- 函数字符串类的主节点 （第 25 页）
- 具有函数字符串类作用域的系统函数 （第 14 页）
- 具有复制定义作用域的系统函数 （第 17 页）

系统函数摘要

Replication Server 提供了具有函数字符串类作用域和复制定义作用域的系统函数。

有关所有系统函数的完整文档，请参见《Replication Server 参考手册》中的“Replication Server 系统函数”。

具有函数字符串类作用域的系统函数

Replication Server 提供了一些具有函数字符串类作用域的系统函数。

在您安装复制系统时，Replication Server 会为每个系统提供的类提供缺省生成的函数字符串。

有些函数是每个 Replication Server 应用程序都需要的，而其它函数则只适用于特定的情况，例如，热备份应用程序、并行 DSI 线程，或协调的转储。

如果您使用的函数字符串类不是缺省的 (**rs_sqlserver_function_class**)，并且没有使用函数字符串继承，则必须为您使用的每个具有函数字符串类作用域的系统函数创建函数字符串。

在作为函数字符串类的主节点的 Replication Server 中，为具有类作用域的系统函数自定义函数字符串。您还可能需要为函数字符串类分配主节点，或将主节点从一个 Replication Server 更改为另一个 Replication Server。

表 3. 具有函数字符串类作用域的系统函数

函数名	说明
rs_batch_start	指定为了标记一批命令的开始除 **rs_begin** 语句之外还需提供的 SQL 语句。
rs_batch_end	指定为标记一批命令结尾所需的 SQL 语句。此函数字符串与 **rs_batch_start** 一起使用。
rs_begin	开始事务。
rs_check_repl	检查表是否标记为要复制。
rs_commit	提交事务。
rs_dumpdb	启动协调的数据库转储。
rs_dumptran	启动协调的事务转储。
rs_get_charset	返回数据服务器使用的字符集。
rs_get_lastcommit	从 `rs_lastcommit` 系统表中检索行。
rs_get_sortorder	返回数据服务器使用的排序顺序。
rs_get_thread_seq	返回 `rs_threads` 系统表中的指定条目的当前序列号。只有在您使用并行 DSI 时，才会执行此函数。
rs_get_thread_seq_noholdlock	使用 **noholdlock** 选项，返回 **rs_threads** 系统表中的指定条目的当前序列号。当 dsi_isolation_level 为 3 时，将使用此线程。
rs_initialize_threads	将 `rs_threads` 系统表中每个条目的序列设置为 0。只有在您使用并行 DSI 时，才会执行此函数。
rs_marker	帮助协调预订实现。该函数将它的第一个参数作为独立命令传递到 Replication Server。
rs_non_blocking_commit	与复制数据服务器中的相应函数协调 Replication Server 非阻塞提交。 映射到 Adaptive Server 15.0 和更高版本中的 **set delayed_commit on** 函数字符串以及 Oracle 10g v2 中的 **alter session set commit_write = nowait;** 函数字符串。对于其它所有非 Sybase 数据库，**rs_non_blocking_commit** 映射到 null。 如果 **dsi_non_blocking_commit** 的值在 1 到 60 之间，每次 DSI 连接至复制数据服务器时，均会执行。如果 **dsi_non_blocking_commit** 的值为零，则 **rs_non_blocking_commit** 不会执行。

函数名	说明
rs_non_blocking_commit_flush	在启用 **dsi_non_blocking_commit** 时，可确保将数据库事务刷新到磁盘。 映射到 Adaptive Server 15.0 和更高版本以及 Oracle 10g v2 和更高版本中的相应函数字符串。对于其它所有非 Sybase 数据库，**rs_non_blocking_commit_flush** 映射到 null。 **rs_non_blocking_commit_flush** 以等于您使用 **dsi_non_blocking_commit** 指定的任意分钟数（1 到 60）的间隔执行。如果 **dsi_non_blocking_commit** 值为零，**rs_non_blocking_commit_flush** 则不执行。
rs_raw_object_serialization	将 Java 列作为序列化数据进行复制。
rs_repl_off	为连接备用数据库而将 Adaptive Server 中的复制设置为“关闭”。
rs_repl_on	为连接备用数据库而将 Adaptive Server 中的复制设置为“打开”。
rs_rollback	回退事务。
rs_set_ciphertext	打开 **set ciphertext on**，将启用对 **rs_default_function_class** 和 **rs_sqlserver_function_class** 的加密列的复制。对于所有其它类，此函数设置为 null。
rs_set_isolation_level	将事务隔离级别传递给复制数据服务器。
rs_set_dml_on_computed	当建立一个连接时，在复制数据库 DSI 上进行应用。它在 **use database** 语句之后发出 **set dml_on_computed “on”** 命令。
rs_set_proxy	假定用户的权限、登录名和服务器用户 ID。
rs_set_quoted_identifiers	设置与数据服务器的 DSI 连接，以允许通过该连接发送带引号的标识符。 前提条件：若要为数据服务器启用带引号的标识符，**dsi_quoted_identifier** 必须设置为“on”，并且 **rs_set_quoted_identifier** 必须包含所需的命令。对于 Adaptive Server 和 Microsoft SQL Server，该命令为：**set quoted_identifiers on**。
rs_thread_check_lock	确定 DSI 执行程序线程是否持有阻塞复制数据库进程的锁。
rs_triggers_reset	为连接备用数据库而在 Adaptive Server 中将触发器设置为“关闭”。
rs_trunc_reset	在热备份数据库中重新设置辅助截断点。只有在您创建热备份数据库或切换到备用数据库时，才会执行此函数。
rs_trunc_set	在热备份数据库中设置辅助截断点。只有在您创建热备份数据库或切换到备用数据库时，才会执行此函数。
rs_update_threads	更新 `rs_threads` 表中的指定条目的序列号。只有在您使用并行 DSI 时，才会执行此函数。

函数名	说明
rs_usedb	更改数据库环境。

另请参见

- 更改函数字符串类的主节点 （第 25 页）

具有复制定义作用域的系统函数

Replication Server 提供了一些具有复制定义作用域的系统函数。

在您创建复制定义时，Replication Server 会为每个系统提供的类提供缺省的函数字符串。

有些函数是每个 Replication Server 应用程序都需要的，而其它函数则只适用于特定情况，例如，复制 `text`、`unitext` 和 `image` 数据类型、并行 DSI 线程，或执行实现预订或取消实现预订。

在创建了复制定义的 Replication Server 上为具有复制定义作用域的系统函数自定义函数字符串。

表 4. 具有复制定义作用域的系统函数

函数名	说明
rs_datarow_for_writetext	提供与 `text`、`unitext` 或 `image` 列关联的数据行的映像，这些列通过 Transact-SQL® **writetext** 命令或者 CT-Library 或 DB-Library™ 函数更新。
rs_delete	在表中删除行。
rs_get_textptr	检索 `text`、`unitext`、`image` 或 `rawobject` 列的文本指针。
rs_insert	在表中插入行。
rs_select	从表中检索行以便实现预订或取消实现预订。
rs_select_with_lock	使用 holdlock 检索实现预订或取消实现预订的行。
rs_textptr_init	为 `text`、`unitext`、`image` 或 `rawobject` 列分配文本指针。
rs_truncate	截断表。
rs_update	更新表中的行。
rs_writetext	更改 `text`、`unitext`、`image` 或 `rawobject` 数据。

函数字符串

函数字符串包含用于执行数据库中函数的指令。

这些指令可能会因数据库而异。例如，与 Adaptive Server 数据库相比，非 Sybase 数据库要求的指令可能会不同，而且可能具有不同的函数字符串。

函数字符串具有两种格式：语言和（远程过程调用）RPC。语言格式函数字符串包含一个数据服务器分析的命令，例如一个 SQL 语句。RPC 格式函数字符串包含一个远程过程调用，该调用会执行 Open Server™ 网关应用程序或 Adaptive Server 数据库中的一个已注册过程。这两种函数字符串格式都可以包含可替换为数据值的变量。函数字符串所用的格式取决于数据服务器的类型以及您希望 Replication Server 与函数字符串如何交互。您可以更改输出模板以自定义函数字符串。

函数字符串划分为函数字符串类的组。必须根据复制数据库的类型，为每个数据库连接指派一个函数字符串类。Replication Server 提供了一些函数字符串类，以便为当前支持的所有数据服务器生成缺省函数字符串。

在设置复制系统或为该系统添加数据库时，应该预计到函数字符串的要求，并确定如何使用函数字符串类以及是否需要自定义函数字符串。

另请参见

- 输出模板 （第 28 页）
- 函数字符串类 （第 19 页）
- 管理函数字符串 （第 27 页）

输入模板和输出模板

每个函数字符串都使用一个输出模板，以指示目标数据库为特定数据服务器执行该函数。

rs_select 和 **rs_select_with_lock** 函数的函数字符串同时使用输入模板和输出模板，这两种模板共同执行实现预订和取消实现预订。

您可以通过更改函数字符串的输入模板和输出模板来自定义函数字符串。对于除 **rs_select** 和 **rs_select_with_lock** 以外的函数，通过只更改输出模板即可为它们自定义函数字符串。如何更改函数字符串取决于函数字符串的格式是语言还是 RPC。

另请参见

- 函数字符串输入模板和输出模板 （第 28 页）

自定义函数字符串的应用

自定义函数字符串具有一些应用场合。

- 通过更改函数字符串输出模板来设置发送到数据服务器的命令的格式，从而使用任何本机数据库语言（包括 Transact-SQL 以外的本机数据库语言）执行操作。
- 使用一个函数字符串实现和取消实现对同一复制定义的多个预订。

- 将现有系统函数字符串的输出模板更改为：
  - 记录审计信息。
  - 执行远程过程调用 (RPC)。
  - 将数据复制到同一数据库的多个复制表中。
  - 将数据复制到与主表具有不同的名称、列名和列顺序的复制表中。
    如果复制 Replication Server 的版本是 11.5 或更高版本，您可以创建自定义复制定义以指定有关复制表的信息，从而可以更轻松地执行相同的任务。请参见《Replication Server 管理指南第一卷》的“管理复制表”的“创建复制定义”中的“每个表创建多个复制定义”。

具有多个函数字符串的系统函数

对于其它具有复制定义作用域的系统函数，您可以为同一复制定义创建多个函数字符串实例。

对于类作用域系统函数，每个函数都会映射到类内的一个函数字符串。每个复制定义作用域 **rs_insert**、**rs_delete** 和 **rs_update** 系统函数都会每个复制定义映射到类内的一个函数字符串。

对于其它具有复制定义作用域的系统函数，您可以为同一复制定义创建多个函数字符串实例：**rs_select**、**rs_select_with_lock**、**rs_datarow_for_writetext**、**rs_get_textptr**、**rs_textptr_init** 和 **rs_writetext**。在这种情况下，您必须为每个函数字符串实例指定不同的名称。可以带有多个函数字符串的系统函数包括：

- **rs_select** 函数和 **rs_select_with_lock** 函数 – 当针对同一复制定义存在多个预订时，用于实现预订和取消实现预订。您可以为每个函数字符串实例指定任意名称，但该名称对于复制定义要是唯一的。每个函数字符串实例都对应一个在为复制定义创建预订时使用的 **where** 子句。
- **rs_datarow_for_writetext**、**rs_get_textptr**、**rs_textptr_init** 和 **rs_writetext** 函数 – 函数字符串的各个实例。您必须命名复制定义中指定的 `text`、`unitext` 或 `image` 列的函数字符串的每个实例。

函数字符串类

每个函数字符串都属于一个函数字符串类，该类将函数字符串分组，以便用于类型或要求相似的数据库。

Replication Server 会根据目标数据库的数据服务器为每个数据库连接指派一个函数字符串类。

Replication Server 会通过从指派给数据库的函数字符串类中使用函数字符串来将函数应用于该数据库。函数字符串类包含用于系统函数的函数字符串以及用于任何用户定义的函数的函数字符串。

如果函数字符串可以在所有数据服务器上执行，您可以在多个数据库上使用一个函数字符串类。例如，一个有若干个由 Adaptive Server 管理的数据库的系统可以将 **rs_sqlserver_function_class** 用于所有数据库。

您甚至可以将一个函数字符串类用于几个非 ASE 数据服务器，但前提是您使用 ECDA 访问各种数据服务器。

系统提供的类

Replication Server 提供了一些函数字符串类（称为系统提供的类），包括用于 Replication Server 支持的数据服务器的缺省函数字符串。

- **rs_sqlserver_function_class** – 为此类提供了缺省的 Adaptive Server 函数字符串。**rs_sqlserver_function_class** 中的缺省函数字符串与 **rs_default_function_class** 中的缺省函数字符串相同。缺省情况下，将 **rs_sqlserver_function_class** 指派给使用 **rs_init** 添加到复制系统中的 Adaptive Server 数据库。
  您可以为此类自定义函数字符串。但是，此类无法参与函数字符串类继承。多数情况下，最好使用将 **rs_default_function_class** 指定为父类的派生类，而不直接使用 **rs_sqlserver_function_class**。
- **rs_default_function_class** – 为此类提供了缺省的 Adaptive Server 函数字符串。**rs_sqlserver_function_class** 中的缺省函数字符串与 **rs_default_function_class** 中的缺省函数字符串相同。
  您无法为此类自定义函数字符串。但是，此类可以参与函数字符串类继承。多数情况下，最好使用将 **rs_default_function_class** 指定为父类的派生类，而不直接使用 **rs_default_function_class**。

  注意： 系统提供的函数字符串类 **rs_default_function_class** 和 **rs_sqlserver_function_class** 包含所有系统函数（**rs_dumpdb** 和 **rs_dumptran** 除外）的缺省函数字符串。如果您需要使用这些函数的函数字符串，必须自己在派生类或 **rs_sqlserver_function_class** 中创建它们。
- **rs_db2_function_class** – 为此类提供了特定于 DB2 的函数字符串。请参见《Replication Server 管理指南第一卷》的“管理复制表”的“使用 HDS 转换数据类型”中的“创建类级别转换”。
  如果您需要 DB2 函数字符串，在多数情况下，最好使用将 **rs_db2_function_class** 指定为父类的派生类，而不直接使用此类。
- **rs_iq_function_class** – 为此类提供了 Sybase® IQ 函数字符串。请参见《Replication Server 管理指南第一卷》的“管理复制表”的“使用 HDS 转换数据类型”中的“创建类级别转换”。
- **rs_msss_function_class** – 为此类提供了 Microsoft SQL Server 函数字符串。请参见《Replication Server 管理指南第一卷》的“管理复制表”的“使用 HDS 转换数据类型”中的“创建类级别转换”。
- **rs_oracle_function_class** – 为此类提供了 Oracle 函数字符串。请参见《Replication Server 管理指南第一卷》的“管理复制表”的“使用 HDS 转换数据类型”中的“创建类级别转换”。

另请参见

- 函数字符串创建准则 （第 34 页）
- 具有函数字符串类作用域的系统函数 （第 14 页）

函数字符串继承

通过创建类之间的关系在类之间共享函数字符串定义的能力被称为“函数字符串继承”。

如果通常使用函数字符串继承，特别是从系统提供的类继承，则会为复制系统管理员带来管理优势和升级优势。如果使用从系统提供的类继承的类，则您只更改要自定义的函数字符串，然后继承所有其它函数字符串。

如果您使用不是从系统提供的类继承的类，则必须自己创建所有函数字符串，而且，只要一新建表或函数复制定义，就要添加新函数字符串。

从父类继承函数字符串的类称为“派生类”。派生类从中继承函数字符串的类称为派生类的“父类”。通常情况下，创建派生类是为了自定义某些特定的函数字符串，并从父类继承所有其它函数字符串。

不从任何父类继承函数字符串的类称为“基类”。系统提供的类 **rs_default_function_class** 和 **rs_db2_function_class** 以及您所创建的不从父类继承函数字符串的任何其它类都是基类。系统提供的类 **rs_iq_function_class**、**rs_msss_function_class** 和 **rs_oracle_function_class** 都是从 **rs_default_function_class** 派生的。

父类可以具有多个派生类，而派生类只能有一个父类。派生类也可以充当一个或多个派生类的父类。源自同一基类的一组任意数量级别的派生类被称为“类树”。

系统提供的类 **rs_default_function_class** 和 **rs_db2_function_class** 可以充当派生类的父类。但是，它们无法成为其它父类的派生类。

系统提供的类 **rs_sqlserver_function_class** 无法充当父类或成为派生类。

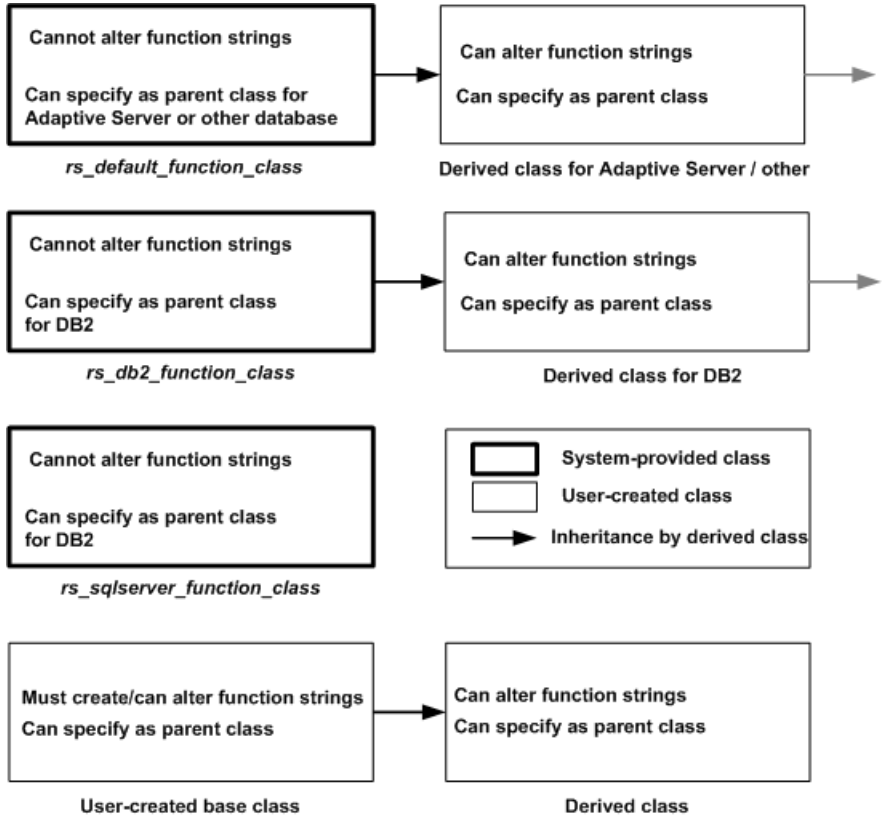
您可以修改所创建的基类，使其成为派生类，或者，可以将它指定为某派生类的父类。可以修改派生类，使其从其它父类继承函数字符串，或者，可以将它与父类分离，使其成为基类。

对于您创建的每个基类，您都必须为 Replication Server 在该类所指派到的每个数据库中调用的函数提供函数字符串。如果您在系统函数的某些函数字符串丢失的情况下将一个函数字符串类指派给数据库，那么，当 Replication Server 尝试应用该函数字符串时，DSI 会报告错误并挂起数据库连接。

不允许循环函数字符串继承关系。即，不能将父类修改为从它自己的派生类中的某一个或这些派生类中的某一个继承函数字符串。

此图说明了函数字符串类关系。

图 1： 函数字符串类关系



混合版本系统中的限制

在混合版本系统中，只有 Replication Server 11.5 版或更高版本可以使用参与函数字符串继承的类。

主节点为 Replication Server 11.0.x 版的任何类都无法参与函数字符串继承。如果您要更改这样的类，使它成为派生类或用它作为父类，必须将该类移至 Replication Server 11.5 或更高版本的主节点。然后，您可以根据需要更改类关系，并将该类或它的派生类指派给由 Replication Server 11.5 或更高版本管理的连接。

您在 Replication Server 11.5 或更高版本中创建的基类以及不参与函数字符串继承的基类可以指派给复制系统中任何 Replication Server 管理的连接。如果没有将它指派给任何由 Replication Server 11.5 或更高版本管理的数据库，您可以使用 **move primary** 命令将它指派给由 Replication Server 11.0.x 版管理的主节点。

有关 Replication Server 之间的兼容性的详细信息，请参见发行公告。

注意： 为了与 Replication Server 11.0.x 版兼容，您可能需要继续自定义 **rs_sqlserver_function_class** 中的函数字符串。但是，对于由 Replication Server 11.5 或

更高版本管理的数据库，建议只在派生类中使用函数字符串继承和自定义函数字符串。

管理函数字符串类

管理函数字符串类包括创建、分配和删除函数字符串类。

您在创建或自定义函数字符串时，会指定它所属的类。如果您要创建和使用自定义函数字符串，可以：

- 创建这样一个派生函数字符串类：它从 **rs_default_function_class**、**rs_db2_function_class** 或其它父类继承函数字符串。然后，在派生类中，只创建您想要替换的函数字符串。

  注意： 您无法更改、添加、删除或更改非 Sybase 数据服务器的任何函数字符串类。
- 新建函数字符串类并为所有函数创建函数字符串。
- 自定义 **rs_sqlserver_function_class** 中的函数字符串。

您在创建自定义函数字符串之前，应事先确定要采用哪些方法中的哪些并相应地设置类。通常情况下，最好自定义派生类中的函数字符串，而不要自定义类 **rs_sqlserver_function_class** 中的函数字符串。要创建和部署从其它类继承函数字符串的派生函数字符串类，您必须使用 Replication Server 11.5 或更高版本。

另请参见

- 管理函数字符串 （第 27 页）

创建函数字符串类

如果现有类中的函数字符串不能满足您对特定数据库连接的需要，并且无法自定义现有类中的函数字符串，则可以新建一个类，以便在其中创建您需要的函数字符串。

请执行以下操作之一：

- 创建派生类 - 从现有父类继承函数字符串的类。
- 创建基类 - 不从其它类继承函数字符串的类。

1. 使用 **create function string class** 创建函数字符串类。

   使用相应的语法：
   - 创建派生类，或者
   - 创建基类。

   新类的名称必须符合标识符的规则。请参见《Replication Server 参考手册》的“主题”中的“标识符”。
2. 使用 **create function string** 为新类创建函数字符串。

- 如果要使用派生类，只需创建要替换的函数字符串，并从指定的父类继承所有其它函数字符串。
- **rs_default_function_class** 类（系统提供的类）不包含 **rs_dumpdb** 和 **rs_dumptran** 函数的缺省函数字符串。如果在从 **rs_default_function_class** 继承的派生类中需要它们，您必须创建它们。
- 如果要创建基类，必须为该类创建所有必要的函数字符串。

3. 如果您要为现有数据库连接准备新函数字符串类，必须先挂起该连接，然后才能使用该新类。

   请参见《Replication Server 管理指南第一卷》的“管理数据库连接”的“更改数据库连接”中的“挂起数据库连接”。
4. 创建或更改数据库连接以分配新类。
5. 如果已经更改了现有数据库连接以使用新类，请恢复该连接。

   请参见《Replication Server 管理指南第一卷》的“管理数据库连接”的“更改数据库连接”中的“挂起数据库连接”。

另请参见

- 函数字符串继承 （第 21 页）
- 创建派生类 （第 24 页）
- 创建基类 （第 25 页）
- 创建函数字符串 （第 33 页）
- 系统提供的类 （第 20 页）
- 将函数字符串类指派给数据库 （第 26 页）

创建派生类

可以使用 **create function string class** 命令并指定父类，以创建从父类继承函数字符串的派生函数字符串类。

例如，在父类的主数据节点上，输入：

```
create function string class
 sqlserver_derived_class
 set parent to rs_default_function_class
```

在此示例中，新类 **sqlserver_derived_class** 从系统提供的类 **rs_default_function_class** 继承函数字符串。然后，您可以创建函数字符串来替换一些继承函数字符串。

您可以将其主节点运行 Replication Server 11.5 版或更高版本的任何现有类指定为父类。但是，不能将系统提供的类 **rs_sqlserver_function_class** 指定为父类。此外，也不能指定会导致循环继承的父类。

如果父类为 **rs_default_function_class** 或非 Sybase 数据服务器的函数字符串类，则可以在任何具有到其它 Replication Server（新类将在该处使用）的路由的 Replication Server 中输入此命令。此节点是派生类以及从它派生的任何新类的主节点。

如果父类是用户创建的类，请在充当父类的主节点的 Replication Server 中输入此命令。此节点是从该父类派生的所有类的主节点。

另请参见

- 函数字符串继承 （第 21 页）

创建基类

可以使用 **create function string class** 命令而不指定父类，以创建不从父类继承函数字符串的基本函数字符串类。

例如，输入：

```
create function string class base_class
```

在此示例中，新类 **base_class** 不从父类继承函数字符串。

在任何具有到其它 Replication Server（新类将在该处使用）的路由的 Replication Server 中输入此命令。这样，此节点将成为该类以及将该类作为父类的任何派生类的主节点。

基类可以用作派生类的父类，也可以经过修改而成为派生类。

对于您创建的每个基类，您都必须为 Replication Server 在该类所指派到的每个数据库中调用的函数提供函数字符串。

如果您创建一个基类，然后，在将它实际用于数据库连接之前更改它，从而使它成为派生类，则您不必创建所有函数字符串。

函数字符串类的主节点

虽然大多数函数字符串都在复制数据库中执行，但应在 Replication Server（通常是主 Replication Server，它具有到所有要使用函数字符串类的节点的路由）中执行 **create function string class** 命令。

此命令将该 Replication Server 指派为该类的主节点。函数字符串类是通过路由与其它复制系统数据一起进行复制的。

您只能在类的主节点中创建或更改具有类作用域的函数字符串。必须在复制定义的主节点中创建或更改具有复制定义作用域的函数字符串。

缺省情况下，类 **rs_sqlserver_function_class** 没有主节点。要更改此类的类作用域函数字符串，必须先将 Replication Server 指派为该类的主节点。要指定此函数字符串类的节点，请在要成为主节点的 Replication Server 中执行以下命令：

```
create function string class rs_sqlserver_function_class
```

执行此命令以后，您可以使用 **move primary** 命令进一步更改该函数字符串类的主节点。

更改函数字符串类的主节点

使用 **move primary** 命令或 Sybase Central 更改函数字符串类的主 Replication Server。

例如，您可能需要将主节点从一个 Replication Server 更改为另一个 Replication Server，以便可以通过新的路由配置分配函数字符串。新的主节点必须包括到将会使用函数字符串类的所有 Replication Server 的路由。

如果移动基类，则从该类派生出的所有类均会随它一起移动。

除非派生类的父类是缺省的函数字符串类，否则不能移动派生类的主节点。

在要指派为函数字符串类的新主节点的 Replication Server 中，执行 **move primary**。

例如，以下命令将 **sqlserver2_function_class** 函数字符串类的主节点更改为 SYDNEY_RS Replication Server，在那里输入命令：

```
move primary of function string class
 sqlserver2_function_class
 to SYDNEY_RS
```

如果还没有为 **rs_sqlserver_function_class** 类指派主节点，您不能使用 **move primary** 为其指派主节点。必须先使用 **create function string class** 为该类指派主节点。

另请参见

- 函数字符串类的主节点 （第 25 页）

将函数字符串类指派给数据库

您可以在 Sybase Central 中为数据库连接指派函数字符串类，或者使用在管理数据库的 Replication Server 中执行的 **create connection** 或 **alter connection** 命令来指派。

在您使用 **rs_init** 程序添加数据库连接时，缺省情况下，会将类 **rs_sqlserver_function_class** 指派给该数据库。

您在更改指派给数据库的函数字符串类之前，必须挂起指向该数据库的连接。**create connection** 和 **alter connection** 的 **set function string class** 子句指定要用于数据库的函数字符串类的名称。

在可以将函数字符串类指派给数据库连接之前：

- 您指定的函数字符串类必须已创建，并且可供 Replication Server 使用。
- 必须在类中创建所有必要的函数字符串。

注意： 在使用连接配置文件创建连接时，连接配置文件将指派函数字符串类。

有关使用 **create connection** 和 **alter connection** 命令以及连接配置文件的详细信息，请参见《Replication Server 管理指南第一卷》的“管理数据库连接”中的“创建数据库连接”以及《Replication Server 管理指南第一卷》的“管理数据库连接”中的“更改数据库连接”。另请参见《Replication Server 参考手册》的“Replication Server 命令”中有关这些命令的参考页。

有关 **rs_init** 的详细信息，请参见适用于您的平台的 Replication Server 安装和配置指南。

新建连接示例

以下命令创建指向由 TOKYO_DS 数据服务器管理的 `pubs2` 数据库的连接：

```
create connection to TOKYO_DS.pubs2
 set error class tokyo_error_class
```

```
set function string class tokyo_func_class
set username pubs2_maint
set password pubs2_maint_pw
```

此命令将 **tokyo_func_class** 函数字符串类指派给数据库连接：

更改现有连接示例

以下命令更改现有的数据库连接，以指定另一函数字符串类：

```
alter connection to TOKYO_DS.pubs2
 set function string class tokyo_func_class2
```

另请参见

- 创建函数字符串 （第 33 页）
- 创建函数字符串类 （第 23 页）

删除函数字符串类

如果您确定不再需要使用创建的函数字符串类，可以使用 **drop function string class** 命令从复制系统中删除这些函数字符串类。

除了系统提供的三个类以及任何当前充当父类的用户创建的类以外，您可以删除任何函数字符串类。您必须先删除使用函数字符串类的所有数据库连接，或者将这些连接更改为使用其它类，然后才能删除该函数字符串类。

删除函数字符串类会删除为该类定义的所有函数字符串，并从 RSSD 中删除对该类的所有引用。

例如，要删除 **tokyo_func_class** 函数字符串类及其所有函数字符串，请在 **isql** 命令行中输入：

```
drop function string class tokyo_func_class
```

在作为该类的主节点的 Replication Server 中输入此命令。

请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**drop function string class**”。

管理函数字符串

每个目标 Replication Server 先使用函数字符串将函数转换为适用于目标数据服务器（如 Adaptive Server）的命令，然后再提交这些命令。

有关 DSI 线程（即在复制 Replication Server 中执行此转换的组件）的详细信息，请参见《Replication Server 管理指南第一卷》中的“Replication Server 技术概述”。

有关完整的命令语法和权限，请参见《Replication Server 参考手册》。

函数字符串和函数字符串类

如果不需要自定义函数字符串，则可以使用系统提供的函数字符串类之一来提供缺省的函数字符串。如果您需要类作用域或复制定义作用域的自定义字符串，必须使用系

统提供的类 **rs_sqlserver_function_class**，在该类中，您可以自定义函数字符串，或者创建派生函数字符串类或基本函数字符串类。

- 如果要在其中执行函数的数据库的连接使用系统提供的函数字符串类、直接或间接地从 **rs_default_function_class** 继承的派生类，或者某个非 Sybase 数据服务器的函数字符串类，则会为每个系统函数和用户定义的函数提供缺省函数字符串。
- 如果连接使用用户创建的基本函数字符串类（该类不继承函数字符串）或继承这种类的派生类，您必须为每个系统函数和用户定义的函数创建函数字符串。如果您希望在基类所有的派生类中都能使用这些字符串，则要在基类中创建它们。

另请参见

- 函数字符串类 （第 19 页）

函数字符串输入模板和输出模板

要自定义函数字符串，您要更改它们的输入和输出模板。

根据函数的不同，函数字符串可能既包括输入模板也包括输出模板、只包括输出模板，或不包括任何模板：

- 对用于预订实现的 **rs_select** 和 **rs_select_with_lock** 函数，Replication Server 使用输入模板查找与预订的 **where** 子句相对应的函数字符串。
- 对于所有函数，Replication Server 都使用输出模板将函数映射到语言命令或在目标服务器中应用 RPC 调用。

使用输入模板和输出模板的要求

在更改模板以定义函数字符串时，需要满足一些要求。

这些要求包括：

- 函数字符串输入模板和输出模板最多只能为 64K 字节。函数字符串输入模板或输出模板中的嵌入变量的替代运行期值的结果不能超过 64K。
- 函数字符串输入模板和输出模板用单引号 (') 分隔。
- 函数字符串变量包含在一对问号 (?) 中。
- 变量名及其修饰符用感叹号 (!) 分隔。

语言输出模板还具有其它相关要求。

另请参见

- 输出模板 （第 28 页）

输出模板

Replication Server 使用输出模板确定发送到数据服务器的命令的格式。您可以更改输出模板以自定义函数字符串。

多数输出模板使用以下三种格式之一：语言、RPC 或 **none**，模板格式与函数字符串本身的格式相对应。

rs_writetext 函数字符串的输出模板可以使用 RPC 格式或其它格式中的一个（**writetext** 或 **none**），但不能使用语言输出模板。

另请参见

- 函数字符串 （第 18 页）
- 使用具有 text、unitext、image 和 rawobject 数据类型的函数字符串 （第 44 页）

语言输出模板

语言输出模板包含数据服务器解释为命令的文本。

Replication Server 用值替换输出模板中嵌入的变量，并将最终生成的语言命令传送到数据服务器以进行处理。

在语言输出模板中，Replication Server 以特殊方式解释某些字符：

- 将两个单引号字符 ('') 解释为一个单引号
- 将两个问号 (??) 解释为一个问号
- 将两个分号 (;;) 解释为一个分号

除了嵌入式变量替代和这些特殊解释外，Replication Server 不会尝试解释语言输出模板的其它内容。

另请参见

- 创建函数字符串 （第 33 页）
- 函数字符串变量 （第 31 页）
- 函数字符串变量格式设置 （第 33 页）

RPC 输出模板

不同于语言输出模板，Replication Server 解释 RPC 输出模板的内容。

它们是使用 Transact-SQL **execute** 命令的格式编写的。Replication Server 会分析该输出模板，构造一个远程过程调用以将其发送到 Adaptive Server、Open Server 网关或 Open Server 应用程序。

RPC 输出模板非常适用于没有语言分析程序的网关或 Open Server。RPC 通常比语言请求更加紧凑，并且，由于不需要数据服务器对它们进行分析，所以效率也更高。因此，即使数据服务器支持语言请求，您也可以选择使用 RPC。

使用 none 参数的输出模板

在使用 **none** 参数标识没有输出命令的类级和表级函数字符串以创建或更改函数字符串时，您可以提高函数字符串效率。Replication Server 不在复制数据库上执行这些函数字符串。

rs_writetext 函数字符串的输出模板

Replication Server 支持三种创建 **rs_writetext** 函数字符串的输出格式：RPC、**none** 和 **writetext**。**writetext** 输出模板只能用于 **rs_writetext** 函数字符串。

另请参见

- 使用具有 text、unitext、image 和 rawobject 数据类型的函数字符串 （第 44 页）

输入模板

输入模板只用于非批量实现和使用 **with purge** 进行的取消实现，在这些情况下，Replication Server 必须选择要在所选表中添加或删除的数据。

rs_select 和 **rs_select_with_lock** 是唯一可以包含输入模板的函数字符串。Replication Server 通过以下方法确定在实现或取消实现期间，将哪个函数字符串用于预订：

- 匹配预订的复制定义
- 将输入模板与预订中使用的 **where** 子句相匹配

rs_select 和 **rs_select_with_lock** 还包含输出模板，以指定实际的选择语句或执行所需实现或取消实现的其它操作。

对于系统提供的类，在您创建复制定义时，Replication Server 会为 **rs_select** 和 **rs_select_with_lock** 函数生成缺省的函数字符串。通常，只有在复制定义存在多个预订时，您才需要自定义这些函数字符串。

rs_select 和 **rs_select_with_lock** 函数的函数字符串最常用于实现。如果您计划多次预订同一复制定义，要在创建预订前先创建函数字符串。有关预订实现的详细信息，请参见《Replication Server 管理指南第一卷》的“管理预订”中的“预订实现方法”。

rs_select 和 **rs_select_with_lock** 的函数字符串还可用于取消实现预订，这种情况会使用创建预订时使用的命令的 **where** 子句。删除预订前，这些函数的函数字符串必须存在。有关取消实现的详细信息，请参见《Replication Server 管理指南第一卷》的“管理预订”的“预订命令”中的“drop subscription 命令”。

输入模板可以包含用户定义的变量，其值来自预订的 **where** 子句中的常量。输入模板中不允许使用任何其它类型的函数字符串变量。同一函数字符串中的输出模板可以引用这些用户定义的变量。

如果您需要自定义输出模板来选择实现数据，则可以在 **rs_select** 或 **rs_select_with_lock** 函数字符串中省略输入模板。这样就会创建一个缺省函数字符串，当其它函数字符串的输入模板都与 **select** 命令不匹配时，该函数字符串会与任何 **select** 语句匹配。

就像使用其它具有复制定义作用域的函数一样，您要在创建复制定义的主 Replication Server 中为 **rs_select** 和 **rs_select_with_lock** 函数创建函数字符串。

确定创建函数字符串的位置

确定在其中创建函数字符串的类。

您在为实现创建 **rs_select** 和 **rs_select_with_lock** 函数字符串时，要在指派给主数据库（用于选择实现数据）连接的函数字符串类中进行创建。如果您要使用批量实现，则不需要为实现创建 **rs_select** 和 **rs_select_with_lock** 函数字符串。

您在为取消实现创建 **rs_select** 和 **rs_select_with_lock** 函数字符串时，要在指派给复制数据库（用于选择要取消实现的数据）连接的函数字符串类中进行创建。如果使用带 **without purge** 选项的 **drop subscription** 删除预订，则取消实现不需要 **rs_select** 和 **rs_select_with_lock** 函数字符串。

rs_select 函数字符串示例

在以下示例中，一个节点通过复制定义 **titles_rep** 预订指定出版商的书名。必须有一个带有输入模板的 **rs_select** 函数字符串，该函数字符串将 pubs2 数据库的 titles 表中的 publisher 列与标识该出版商的用户定义值进行比较。

create function string 命令会创建带输入模板的函数字符串，以将 publisher 列 pub_id 与用户定义的变量 ?*pub_id*!*user*? 进行比较。

输入模板将任何预订与 **where pub_id =** *constant* 格式的 **where** 子句匹配。因此，当使用输出模板时，该模板包括 *constant* 值。输出模板从两个不同的表中选择实现数据。

```
create function string titles_rep.rs_select;pub_id
    for sqlserver2_function_class
scan 'select * from titles where pub_id =
    ?pub_id!user?'
output language
    'select * from titles where pub_id =
    ?pub_id!user?
    union
    select * from titles.pending where pub_id =
    ?pub_id!user?'
```

有关完整语法，请参见《Replication Server 参考手册》。

另请参见

- 函数字符串变量 （第 31 页）
- 创建函数字符串 （第 33 页）

函数字符串变量

可以将函数字符串输入模板或输出模板中嵌入的变量作为各种运行期值的符号标记。

变量可以表示列名、系统定义的变量名、用户定义的函数中的参数名，或在输入模板中定义的用户定义变量。变量必须引用一个与指派给它的任何内容具有相同数据类型的值。

函数字符串变量包含在一对问号 (?) 中，如下所示：

```
?variable!modifier?
```

变量的 *modifier* 部分标识变量表示的数据类型。修饰符与变量名之间使用惊叹号 (!) 分隔。

rs_truncate 函数字符串允许基于位置的函数字符串在格式上发生变化：

```
?n!param?
```

其中，*n* 是从 1 到 255 之间的数字，表示函数参数在 LTL 中的位置。LTL 中的 **rs_truncate** 的第一个参数在函数字符串中表示为 `?1!param?`。对于基于位置的函数字符串变量，仅接受修饰符 param。

带有基于位置的变量的 **rs_truncate** 的示例函数字符串如下所示：

```
truncate table publishers partition ?1!param?
```

另请参见

- 缺省系统变量 （第 43 页）

函数字符串变量修饰符

Replication Server 可识别一些函数字符串变量修饰符。

表 5. 函数字符串变量修饰符

修饰符	说明
`new`、`new_raw`	对 Replication Server 要插入或更新的一行中某列的新值的引用。
`old`、`old_raw`	对 Replication Server 要插入或更新的一行中某列的旧值的引用。
`user`、`user_raw`	对 **rs_select** 或 **rs_select_with_lock** 函数字符串的输入模板中定义的变量的引用。
`sys`、`sys_raw`	对系统定义的变量的引用。
`param`、`param_raw`	对存储过程参数的引用。
`text_status`	对 `text`、`unitext` 或 `image` 数据的 `text_status` 值的引用。可能的值有： • 0x000 – 文本字段包含 NULL 值，且尚未初始化文本指针。 • 0x0002 – 已初始化文本指针。 • 0x0004 – 后面为实际文本数据。 • 0x0008 – 后面无文本数据，因为未复制文本数据。 • 0x0010 - 文本数据未被复制，但包含 NULL 值。

注意： 用户定义的函数的函数字符串不能使用 **new** 或 **old** 修饰符。

有关可在函数字符串输入模板或输出模板中使用的系统定义的变量的列表，请参见《Replication Server 参考手册》的“Replication Server 命令”中的“create function string”。

函数字符串变量格式设置

在 Replication Server 将函数字符串输出模板映射到数据服务器命令时，它会使用 Adaptive Server 格式设置变量格式。

对于大多数变量（那些修饰符以 *_raw* 结尾的特殊情况除外），Replication Server 都会按照以下方式设置数据格式：

- 向字符和日期/时间值中出现的单引号字符添加一个额外的单引号。
- 如果字符和日期/时间值两侧缺少单引号字符，请添加。
- 给货币数据类型的值添加相应的货币符号（例如，美元符号）。
- 给二进制数据类型的值添加“0x”前缀。
- 在字符值中现有的反斜杠 (\) 和换行符的实例之间添加一对反斜杠和换行符。Adaptive Server 将后面跟有换行符的反斜杠视为延续字符，因此，会删除添加的字符对，而原有字符保持不变。

对于以 *_raw* 结尾的修饰符，Replication Server 不会以这些方式更改数据类型。

创建函数字符串

可以使用 **create function string** 命令将函数字符串添加到函数字符串类中，或者为目标数据库添加函数字符串。

在函数字符串的主节点中输入函数字符串命令。对于具有以下内容的函数字符串：

- 复制定义作用域 – 主节点是创建复制定义的 Replication Server。
- 类作用域 – 主节点是充当类的主节点的 Replication Server。派生类的主节点与其父类的主节点相同，除非父类是系统提供的类之一。
- 目标作用域函数字符串 – 主节点是控制目标数据库（备用或复制数据库）的 Replication Server。可以针对备用或复制数据库创建目标作用域函数字符串，并且目标作用域函数字符串与任何函数字符串类无关联。

如果您使用的派生函数字符串类的父类不是由系统提供的，那么，您可以选择在父类（而不是实际指派给特定数据库连接的派生类）中自定义函数字符串。这样，该父类的任何其它派生类都可以使用自定义的函数字符串。

另请参见

- 函数字符串类的主节点 （第 25 页）

目标作用域和复制定义作用域的函数字符串的比较

目标作用域和复制定义作用域的函数字符串之间存在多项差异。

复制定义作用域	目标作用域
复制定义作用域的函数字符串与函数字符串类相关联。	目标作用域的函数字符串与目标数据库（备用数据库或复制数据库）相关联。
在管理复制定义的 Replication Server 上按照函数字符串类的复制定义创建复制定义作用域的函数字符串。	在管理备用数据库或复制数据库的 Replication Server 上创建目标表或存储过程的目标作用域函数字符串。
在按照复制定义创建或更改函数字符串时，Replication Server 检查函数字符串的列或参数是否有效。	Replication Server 在使用函数字符串模板转换 DML 命令并将该命令应用于备用数据库或复制数据库之前，Replication Server 不会检查函数字符串的列或参数的有效性。如果列或参数信息不正确，DSI 连接会关闭。您可以在更正函数字符串后恢复 DSI 连接。
在创建复制定义作用域的函数字符串时，Replication Server 为复制定义创建全套缺省函数字符串。	在创建目标作用域的函数字符串时，Replication Server 只创建您指定的函数字符串。 唯一的例外是 **rs_get_textptr** 和 **rs_writetext** 函数字符串，因为它们共存。如果 **rs_writetext** 有自定义函数字符串，则也存在 **rs_get_textptr** 的函数字符串。它们都可以作为自定义函数字符串或作为系统创建的缺省函数字符串而存在。

函数字符串创建准则

在创建函数字符串时，应遵循一些准则。

以下函数字符串的创建准则适用于函数字符串类：

- 如果您需要自定义函数字符串，则可以在系统提供的类 **rs_default_function_class** 和 **rs_db2_function_class** 以外的任何类中进行自定义。
  对于 **rs_db2_function_class**、**rs_iq_function_class**、**rs_msss_function_class** 和 **rs_oracle_function_class**：
  - 无法使用函数字符串类作用域系统函数（如 **rs_begin**）创建自定义类级函数字符串
  - 可以使用复制定义作用域系统函数（如 **rs_insert**）创建自定义表级函数字符串
- 必须先为函数字符串类指派主节点，然后才可以为该类创建函数字符串。系统提供的类 **rs_sqlserver_function_class** 没有主节点，除非您使用 **create function string class** 命令指派一个。
- 如果函数字符串类是新的基类，则必须先为所有必要的系统函数创建函数字符串，然后才能使用该类。

以下准则适用于函数字符串本身：

- 您可以为函数字符串指定一个可选名称。对于 **rs_select**、**rs_select_with_lock**、**rs_datarow_for_writetext**、**rs_get_textptr**、**rs_textptr_init** 和 **rs_writetext** 函数，Replication Server 使用函数字符串名称唯一标识函数字符串。函数字符串名称在您完全限定它们时是唯一的。
- 如果 **rs_select** 或 **rs_select_with_lock** 函数字符串省略了输入模板，则 Replication Server 会匹配任何没有匹配的函数字符串的预订。
- 如果您要为具有复制定义作用域的函数自定义函数字符串，必须先创建函数字符串，然后再创建预订。
- 目标作用域函数字符串没有函数字符串类。在控制备用或复制数据库的 Replication Server 中，使用 **create function string** 针对这些目标数据库创建目标作用域函数字符串。
- 对于备用表或复制表的目标作用域函数字符串，有效函数为：**rs_insert**、**rs_update**、**rs_delete**、**rs_truncate**、**rs_writetext**、**rs_datarow_for_writetext**、**rs_textptr_init** 和 **rs_get_textptr**。
- 您可以将多个命令放置在一个语言输出模板中，用分号分隔它们。
- 可以对非 ASE 服务器进行命令批处理。
  确保已将数据库连接 **batch** 参数设置为允许命令批处理。请参见《Replication Server 管理指南第一卷》的“管理数据库连接”的“更改数据库连接”的“设置和更改影响物理连接的参数”中的“更改影响单个连接的参数”。
- 您可以使用 Adaptive Server 语法为函数字符串中的 *constant* 指定一个空值。
- 在使用 **none** 参数标识没有输出命令的类级和表级函数字符串以创建或更改函数字符串时，您可以提高函数字符串效率。Replication Server 不在复制数据库上执行这些函数字符串。

有关完整语法，请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**create function string**”。

另请参见

- 在函数字符串中定义多个命令 （第 39 页）
- 非 ASE 服务器的命令批处理 （第 40 页）

创建函数字符串示例

使用这些示例创建函数字符串。

rs_begin 函数字符串

为 **rs_begin** 函数创建函数字符串，以通过执行名为 **begin_xact** 的存储过程来开始数据库中的事务：

```
create function string rs_begin
 for gateway_func_class
 output rpc 'execute begin_xact'
```

rs_insert 函数字符串

为引用 **publishers_rep** 复制定义的 **rs_insert** 函数创建函数字符串，在主表中执行 **insert** 后，该函数会在复制数据库中执行 RPC。只在复制数据库中定义存储过程 **insert_publisher**。

```
create function string publishers_rep.rs_insert
    for function class rs_sqlserver_function_class
    output rpc
    'execute insert_publisher
      @pub_id = ?pub_id!new?,
      @pub_name = ?pub_name!new?,
      @city = ?city!new?,
      @state = ?state!new?'
```

upd_bits 目标作用域函数字符串

为 NY_DS 数据服务器的 rdb1 目标数据库中的 **upd_bits** 存储过程创建自定义函数字符串。存储过程的函数具有与存储过程相同的名称。

```
create function string upd_bits.upd_bits
     for database NY_DS.rdb1
     with overwrite
     output language
     'exec upd_bits
           @firstbit = ?firstbit!param?,
           @secondbit = ?secondbit!param?,
     @commit = ?comment!param?'
```

更改函数字符串

alter function string 命令替换现有的函数字符串。

alter function string 与 **create function string** 基本相同，只是前者先要执行 **drop function string** 命令。该函数字符串会在一个事务中被删除并重新创建，以防止因缺少函数字符串而发生错误。

您可以使用 **alter function string** 命令或 **create function string** 命令更改函数字符串。要使用 **create function string** 命令更改函数字符串，必须在函数字符串名称后面包括可选子句 **with overwrite**。此命令会删除并重新创建一个现有的函数字符串，这一点与 **alter function string** 命令相同。

要使用 **alter function string** 命令更改函数字符串，必须首先创建函数字符串。

在派生类中，首先使用 **create function string** 命令替换从父类继承的函数字符串。您无法在派生类中更改函数字符串，除非已经明确地为该派生类创建了该函数字符串。

您要在作为现有函数字符串主节点的 Replication Server 上更改函数字符串。对于以下内容的函数：

- 复制定义作用域 - 在定义了复制定义的主 Replication Server 上更改函数字符串。
- 类作用域 - 在函数字符串类的主节点上更改函数字符串。派生类的主节点与其父类的主节点相同，除非父类是系统提供的类之一。

- 目标作用域 – 在控制备用或复制数据库的 Replication Server 中更改函数字符串。

对于允许多个函数字符串映射的系统函数（例如，**rs_select** 和 **rs_select_with_lock**），要在 **alter function string** 语法中提供完整的函数字符串名。Replication Server 使用该名称确定要更改哪个函数字符串。

有关完整语法，请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**alter function string**”。

另请参见

- 函数字符串类的主节点 （第 25 页）
- 创建函数字符串 （第 33 页）

删除函数字符串

要放弃派生类中的自定义函数字符串，然后从父类中恢复该函数字符串，请删除该函数字符串。

可以使用 **drop function string** 命令删除函数字符串类中的函数字符串，或删除备用和复制数据库的函数字符串。

警告！ 如果要删除并重新创建函数字符串，请使用 **alter function string** 将现有的函数字符串替换为新的函数字符串。使用其它方法删除并重新创建函数字符串，可能会导致出现该函数字符串暂时丢失的情况。在函数字符串被删除之后到重新创建它之前的这段时间内，如果发生一个使用该函数字符串的事务，Replication Server 会将该函数字符串检测为丢失并导致该事务失败。

如果您从派生类删除函数字符串，则要从父类恢复该函数字符串。

请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**drop function string**”。

您也可以从系统提供的类 **rs_sqlserver_function_class** 中删除自定义函数字符串。

要为具有复制定义作用域的函数字符串恢复已删除的缺省函数字符串，请使用 **alter function string** 命令忽略 **output** 子句。

另请参见

- 恢复缺省函数字符串 （第 38 页）

删除函数字符串示例

从这些示例中了解如何删除函数字符串。

删除复制定义的函数字符串

以下命令删除 `sqlserver2_func_class` 类中的 `publishers_rep` 复制定义的 **rs_insert** 函数字符串：

```
drop function string
publishers_rep.rs_insert
for sqlserver2_func_class
```

删除复制定义的函数字符串实例

以下命令删除 `derived_class` 类中的 `publishers_rep` 复制定义的 **rs_select** 函数的函数字符串的 `pub_id` 实例。删除 **rs_select_with_lock** 函数的函数字符串的方式相似。

```
drop function string
publishers_rep.rs_select;pub_id
for derived_class
```

从函数字符串类中删除函数字符串

以下命令从 `gateway_func_class` 函数字符串类中删除 **rs_begin** 函数字符串：

```
drop function string rs_begin
for gateway_func_class
```

删除目标数据库的函数字符串

以下命令从 NY_DS.rdb1 数据库中删除目标存储过程的函数字符串：

```
drop function string upd_bits.upd_bits
for database NY_DS.rdb1
```

恢复缺省函数字符串

要为具有复制定义作用域的系统函数恢复 Adaptive Server 缺省函数字符串，请在 **create function string** 或 **alter function string** 命令中省略 **output** 子句。

对于具有函数字符串类作用域的系统函数，虽然可以指定空模板，但是不能省略输出模板。

有关这些命令的详细信息，请参见《Replication Server 参考手册》中的“Replication Server 命令”。

在所有类（甚至在派生类）中，执行不带 **output** 子句的 **create function string** 或 **alter function string** 命令可以恢复缺省情况下为系统提供的类 **rs_sqlserver_function_class** 和 **rs_default_function_class** 提供的同一函数字符串。

对于将该类指派给的数据库，此方法生成的缺省函数字符串定义可能适用，也可能不适用。如果您要使用自定义的 **rs_sqlserver_function_class**，或者，如果您要为 Adaptive Server 数据库使用其它由用户创建的基类，此方法最为有用。

在派生类中，如果您想放弃某个自定义函数字符串，然后从父类恢复该函数字符串，请删除该函数字符串。

更改函数字符串示例

以下命令将 `publishers_rep` 复制定义的自定义 **rs_insert** 函数字符串替换为缺省函数字符串：

```
alter function string publishers_rep.rs_insert
for rs_sqlserver_function_class
```

在派生类中创建函数字符串示例

您可以在派生函数字符串类中使用此方法将继承的函数字符串替换为 Adaptive Server 缺省函数字符串。

以下命令将 **publishers_rep** 复制定义的继承 `rs_insert` 函数字符串替换为缺省函数定符串：

```
create function string publishers_rep.rs_insert
for derived_class
```

另请参见

- 删除函数字符串 （第 37 页）
- 更改函数字符串 （第 36 页）
- 创建函数字符串 （第 33 页）

使用输出模板创建空函数字符串

您可以创建一个不执行任何操作的空函数字符串。方法是：包括 **output language** 子句，在其中用两个单引号指定空函数字符串。

例如，以下命令不会为 `publishers_rep` 复制定义的 **rs_insert** 函数字符串定义任何操作：

```
alter function string publishers_rep.rs_insert
for derived_class
output none
```

另请参见

- 更改函数字符串 （第 36 页）

在函数字符串中定义多个命令

您可以使用函数字符串对数据库服务器的命令进行批处理。

语言输出模板可以包含许多命令。Adaptive Server 允许一批中包含多个命令。尽管大多数其它数据服务器都不提供此功能，但 Replication Server 允许您通过用分号 (;) 分隔命令来为任何数据服务器批处理函数字符串中的命令。

使用两个连续的分号 (;;) 表示不 被解释为命令分隔符的一个分号。

如果数据服务器支持命令批处理，Replication Server 会根据需要用 DSI 命令分隔符（**dsi_cmd_separator** 配置参数）替换分号，并在一批中提交这些命令。

如果数据服务器不支持命令批处理，Replication Server 会分别提交函数字符串中的每个命令。

例如，以下函数字符串中的输出模板包含两个命令：

```
create function string rs_commit
for sqlserver2_function_class
```

```
output language
'execute rs_update_lastcommit
    @origin = ?rs_origin!sys?,
    @origin_qid = ?rs_origin_qid!sys?,
    @secondary_qid = ?rs_secondary_qid!sys?;
    commit transaction'
```

在 Replication Server 中，使用 **alter connection** 命令可启用或禁用对批处理的支持。

如果将 **batch** 设置为“on”，则允许对数据库进行命令批处理；或者，如果设置为“off”，则将单个命令发送到数据服务器。

在本示例中，若要将批处理设置为“on”，请输入：

```
alter connection to SYDNEY_DS.pubs2
    set batch to 'on'
```

若要将批处理设置为“off”，请输入：

```
alter connection to SYDNEY_DS.pubs2
    set batch to 'off'
```

非 ASE 服务器的命令批处理

Replication Server 允许您对非 ASE 数据库服务器的命令进行批处理，这可能会提高性能。

支持命令批处理需要：

- 使用两个函数字符串，即 **rs_batch_start** 和 **rs_batch_end**。
- 使用 DSI 连接参数控制对两个函数字符串的处理。

支持命令批处理的函数字符串

对非 ASE 服务器的命令批处理支持是通过使用 **rs_batch_start** 和 **rs_batch_end** 这两个函数字符串实现的。

这些函数字符串存储了用于标记命令批处理的开始和结束所需的 SQL 转换。对于 ASE 或函数字符串 **rs_begin** 和 **rs_commit** 已经支持所需功能的任何其它数据服务器来说，无需使用这些函数字符串

支持命令批处理的连接设置

use_batch_markers DSI 连接参数用于控制对 **rs_batch_start** 和 **rs_batch_end** 函数字符串的处理。

可以使用 **alter connection** 和 **configure connection** 设置 **use_batch_markers**。如果 **use_batch_markers** 设置为 on，则会执行 **rs_batch_start** 和 **rs_batch_end** 函数字符串。缺省值为 off。

注意： 如果复制数据服务器需要在批处理命令的开头或结尾处发送其它 SQL，而该 SQL 语句未包含在 **rs_begin** 函数字符串中，此时才需要将 **use_batch_markers** 设置为 on。

处理顺序

当您将连接配置为使用批处理标记函数字符串时，Replication Server 按特定顺序将语句发送到数据服务器。

1. 首先基于配置参数 **batch_begin** 将 **rs_begin** 命令单独或与这批命令一起发送到复制数据服务器。
2. 只有在将 **use_batch_markers** 配置为 true 时，才会处理并发送 **rs_batch_start** 命令。

   将 **rs_batch_start** 标记与正在作为一个批次发送的命令组合在一起。有效的 **rs_begin** 和 **rs_batch_start** 函数字符串既支持发送给数据服务器的单个事务的处理，又支持成批事务的处理。
3. 将一批命令发送到复制数据服务器。

   批次大小会改变，命令批的发送则遵循对发送到复制数据服务器的命令的分组和刷新予以终止的现有规则。这些命令的各个命令之间都包含一个命令分隔符。
4. **rs_batch_end** 命令是命令批中的最后一个命令。只有在将 **use_batch_markers** 配置参数设置为 true 时，才会发送 **rs_batch_end** 标记。

   如果在以 **dsi_cmd_batch_size** 等限制刷新命令后需要多批命令，则可能重复 **rs_batch_start**、一批命令和 **rs_batch_end**。
5. 发送最后的 **rs_batch_end** 命令后，**rs_commit** 命令将发送到复制数据服务器。根据现有规则处理 **rs_commit**。

DSI 配置

对于将作为批处理命令的各个连接，需要考虑以下一些 DSI 配置参数：

- **batch**
- **batch_begin**
- **use_batch_markers**

若要确定是否允许对非 ASE 复制数据服务器进行命令批处理，请参见《Replication Server 异构复制指南》。

要使用这些配置参数，请参见《Replication Server 管理指南第一卷》的“管理数据库连接”的“更改数据库连接”的“设置和更改影响物理连接的参数”中的“影响物理数据库连接的配置参数”以及《Replication Server 参考手册》的“Replication Server 命令”中的“**alter connection**”。

在语言输出模板中使用声明语句

若要在语言输出模板中包括用于定义局部变量的声明语句，一定要将连接到数据库的 Replication Server 的 **batch** 配置参数设置为“off”。

如果将 **batch** 设置为“on”（Adaptive Server 缺省值），则 Replication Server 可以将某个函数字符串的多个调用作为一批命令发送到数据服务器，从而在该批命令中加入同一变量的多个声明，但 Adaptive Server 不允许这样做。

如果关闭批处理模式，性能会降低，原因是，Replication Server 必须等待每个命令得到响应之后，才会发送下一个命令。如果对性能的要求较低，在将 **batch** 设置为“off”

的情况下，您可以在函数字符串中使用声明语句。或者，如果要使用批处理模式以提高性能，可创建执行存储过程的函数字符串语言输出模板，其中可以包括声明语句和其它命令。

有关 **batch** 的详细信息，请参见《Replication Server 管理指南第一卷》的“管理数据库连接”的“更改数据库连接”的“设置和更改影响物理连接的参数”中的“影响物理数据库连接的配置参数”。

显示与函数相关的信息

可以使用 Replication Server **admin** 命令或 Adaptive Server 存储过程获取有关复制系统中的现有函数字符串和类的信息。

有关 **admin** 命令的详细信息，请参见《Replication Server 参考手册》中的“Replication Server 命令”。

使用 admin 命令获取信息

可以使用某个 Replication Server **admin** 命令显示 Replication Server 系统中使用的函数字符串类的名称。

使用 **admin show_function_classes** 可以显示现有函数字符串类及其父类的名称，它还可以指明类的继承级别。级别 0 是基类（例如 **rs_default_function_class** 或 **rs_db2_function_class**），级别 1 是从基类继承的派生类，依此类推。

例如：

```
admin show_function_classes
```

```
  Class                  ParentClass                  Level
  --------               ------------                 -----
   sql_derived_class       rs_default_function_class     1
   rs_db2_derived_class    rs_db2_function_class         1
   rs_db2_function_class                                 0
...
```

请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**admin show_function_classes**”。

使用存储过程获取信息

使用 Replication Server 的 RSSD 中的存储过程，可以获取有关系统中的现有函数、函数字符串和函数字符串类的信息。

请关详细信息，请参见《Replication Server 参考手册》中的“RSSD 存储过程”。

rs_helpfunc

rs_helpfunc 显示有关 Replication Server 或者特定表或函数复制定义的系统函数和用户定义函数的信息。语法为：

```
rs_helpfunc [replication_definition [, function_name]]
```

rs_helpfstring

rs_helpfstring 显示与复制定义相关联的函数的参数和函数字符串文本。语法为：

```
rs_helpfstring replication_definition
    [, function_name]
```

rs_helpobjfstring

rs_helpobjfstring 显示与备用复制表或存储过程关联的函数字符串的参数和函数字符串文本。语法为：

```
rs_helpobjfstring data_server, database, [owner.]object [,function]
```

rs_helpclass

rs_helpclass 列出了所有函数字符串类和错误类及其主 Replication Server。语法为：

```
rs_helpclass [class_name]
```

rs_helpclassfstring

rs_helpclassfstring 显示了类作用域函数的函数字符串信息。语法为：

```
rs_helpclassfstring class_name [, function_name]
```

缺省系统变量

可以使用缺省系统变量 *rs_default_fs* 扩展和自定义函数字符串。

- 扩展具有复制定义作用域的函数字符串，以包括其它命令（例如用于审计或跟踪的命令）
- 自定义 **rs_update** 和 **rs_delete** 函数字符串，并且仍能在复制定义中使用 **replicate minimal columns** 选项

注意： 包含 *rs_default_fs* 系统变量的函数字符串只能应用于 Adaptive Server 或接受 Adaptive Server 语法的数据服务器。否则，将会发生错误。

有关函数字符串系统变量的完整列表，请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**create function string**”。

扩展缺省函数字符串

您可以将 *rs_default_fs* 系统变量用于所有具有复制定义作用域（表或函数）的函数字符串，作为扩展缺省函数字符串的行为的一种方法。

如果您想保持缺省字符串的功能不变，并包括其它命令，使用 *rs_default_fs* 系统变量可减少所需的键入量。例如，可以添加命令来扩展缺省函数字符串的审计或跟踪功能。

添加到输出语言模板的命令可以在系统变量 *rs_default_fs* 之前或之后。它们可能会也可能不会影响将行复制到复制表中的方式。

以下示例显示如何在 **create function string** 命令（或 **alter function string** 命令）中使用 *rs_default_fs* 系统变量来验证是否进行了更新：

```
create function string replication_definition.rs_update
  for function_string_class
  output language '?rs_default_fs!sys?;
if (@@rowcount = 0)
  begin
  raiserror 99999 "No rows updated!"
  end'
```

在此示例中，嵌入在语言输出模板中的 *rs_default_fs* 系统变量保持缺省 **rs_update** 函数字符串的功能，而输出模板则检查是否更新了任何行。如果未更新这些行，则会引发错误。

在此示例中，系统变量后面的命令不会影响在复制节点复制行的方式。您可以在其它相似命令中使用 *rs_default_fs* 系统变量来进行验证或审计。

使用 replicate minimal columns 子句

自定义 **rs_update** 和 **rs_delete** 函数字符串，并继续使用 **replicate minimal columns** 子句。

如果已经为复制定义指定了 **replicate minimal columns** 子句，则通常无法为 **rs_update**、**rs_delete**、**rs_get_textptr**、**rs_textptr_init** 或 **rs_datarow_for_writetext** 系统函数创建非缺省函数字符串。

通过在 **create function string** 或 **alter function string** 命令的输出语言模板中嵌入 *rs_default_fs* 系统变量并仍旧使用最少列选项，可为 **rs_update** 和 **rs_delete** 函数创建非缺省函数字符串。

在使用最少列选项的复制定义的 *rs_update* 或 **rs_delete** 函数字符串中，不能使用任何访问非键列值的变量，包括 **rs_default_fs** 系统变量。创建这种函数字符串时，您可能事先不知道要修改主表中的哪些列。但是，您可以包括访问键列值的变量。

有关 **replicate minimal columns** 子句的详细信息，请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**create replication definition**”。

使用具有 text、unitext、image 和 rawobject 数据类型的函数字符串

在支持 `text`、`unitext`、`image` 和 `rawobject` 数据类型的环境中，可使用输出模板格式 **writetext** 或 **none** 为 **rs_writetext** 函数自定义函数字符串。

请参见《Replication Server 参考手册》的“Replication Server 系统函数”中的“**rs_writetext**”。

对于 Replication Server 11.5 版或更高版本，可以使用多个复制定义来代替函数字符串。请参见《Replication Server 管理指南第一卷》的“管理复制表”的“创建复制定义”中的“每个表创建多个复制定义”。

使用 rs_writetext 函数字符串的 writetext 输出模板选项

rs_writetext 函数字符串的 **writetext** 输出模板选项指示 Replication Server 使用 Client-Library™ 函数 **ct_send_data** 更新 text、unitext、image 或 rawobject 列的值。

该选项指定复制数据库中 text、unitext、image 和 rawobject 列的记录行为。

writetext 输出模板支持以下选项：

- **use primary log** – 在复制数据库中记录数据（如果在主数据库中指定了记录选项）。
- **with log** – 在复制数据库事务日志中记录数据。
- **no log** – 不会将数据记录到复制数据库事务日志中。

使用 rs_writetext 函数字符串的 none 输出模板

rs_writetext 函数字符串的 **none** 输出模板选项指示 Replication Server 不要复制 text、unitext 或 image 列值，从而为在异构环境中使用 text、unitext 和 image 列提供了必要的灵活性。

异构复制与 text、unitext、image 和 rawobject 数据

若要将 text、unitext、image 和 rawobject 数据从非 ASE 数据服务器复制到 Adaptive Server 数据库中，必须在复制定义中包含 text、unitext、image 和 rawobject 数据，以便为 Adaptive Server 数据库创建预订。

但是，您可能不想将 text、 unitext、image 和 rawobject 数据复制到其它复制数据服务器中，无论它们是其它外部数据服务器还是其它 Adaptive Server。

使用 **none** 输出模板选项，可以自定义 **rs_writetext** 函数字符串，以将操作映射到复制节点中较小的表，并指示 **rs_writetext** 函数字符串不要对复制节点执行任何 text、unitext、image 或 rawobject 操作。

复制定义中的每个 **text**、unitext、image 和 rawobject 列都具有一个 rs_writetext 函数字符串。如果您不想复制某个 text、unitext、image 或 rawobject 列，请为该列自定义 **rs_writetext** 函数字符串。在 **create** 或 **alter function string** 命令中指定列名，如下例所示。此外，您可能还需要自定义 **rs_insert** 函数字符串。

示例

假定复制定义不允许在 text、unitext、image 或 rawobject 列中使用空值，并且您在复制节点上不需要某些 text、unitext、image 或 rawobject 列。

如果在主节点的那些列中发生插入，您就必须为在复制节点不需要的 **text**、unitext、image 或 rawobject 列自定义 rs_writetext 函数字符串。您还必须为复制定义自定义 **rs_insert** 函数字符串。

例如，假定您具有主表 foo：

```
foo (int a, b text not null, c image not null)
```

在 foo 中，执行以下插入：

```
insert foo values (1, "111111", 0x11111111)
```

缺省情况下，Replication Server 将 **rs_insert** 转换为以下形式，以便由 DSI 线程应用于复制表 foo：

```
insert foo (a, b, c) values (1, "", "")
```

DSI 线程调用：

- **ct_send_data**，以将 text 数据插入列 b
- **ct_send_data**，以将 image 数据插入列 c

由于 text 列 b 和 image 列 c 不允许空值，所以，如果复制表不包含列 b 或列 c，DSI 线程就会关闭。

如果复制表只包含 a 和 b 列，您需要为 c 列自定义 **rs_writetext** 函数，以便使用 **output none**，如下所示：

```
alter function string foo_repdef.rs_writetext;c
    for rs_sqlserver_function_class
    output none
```

您必须按如上所示指定列名（在本例中为 c），以更改该列的 **rs_writetext** 函数字符串。

如果复制表只包含 a 和 b 列，您还需要为复制定义自定义 **rs_insert** 函数字符串，这样它就不会尝试插入 c 列，如下所示：

```
alter function string foo_repdef.rs_insert
    for rs_sqlserver_function_class
    output language
    'insert foo (a, b) values (?a!new?, "")'
```

如果复制定义指定列 **c** 允许空值，则不必自定义 rs_insert。缺省情况下，**rs_insert** 不影响任何允许空值的 text、unitext 或 image 列。

管理热备份应用程序

在两个数据库（主数据库或活动数据库和单个备用数据库）之间设置、配置和监控热备份应用程序。

对主数据库的更改将直接复制到热备份数据库。要更改或限定发送的数据，必须添加表和函数复制定义。

Replication Server 支持为 Adaptive Server 和 Oracle 数据库设置和管理热备份应用程序。有关如何在两个 Oracle 数据库之间设置和配置热备份应用程序的详细信息，请参见《Replication Server 异构指南》中的“Oracle 异构热备份”。

还可以使用多节点可用性 (MSA) 来设置 Adaptive Server 数据库之间的热备份应用程序。MSA 允许复制到多个备用数据库和复制数据库。您可以选择复制整个数据库，或者复制（或不复制）指定的表、事务、函数、系统存储过程和数据定义语言 (DDL)。请参见《Replication Server 管理指南第一卷》中的“使用多节点可用性管理复制对象”。

热备份应用程序

热备份应用程序是一对数据库，其中一个作为另一个的备份副本。客户端应用程序将更新活动数据库；Replication Server 将维护作为活动数据库副本的备用数据库。

如果活动数据库出现故障，或者需要对活动数据库或数据服务器执行维护操作，则可切换到备用数据库，这样客户端应用程序就可在几乎没有中断的情况下恢复工作。

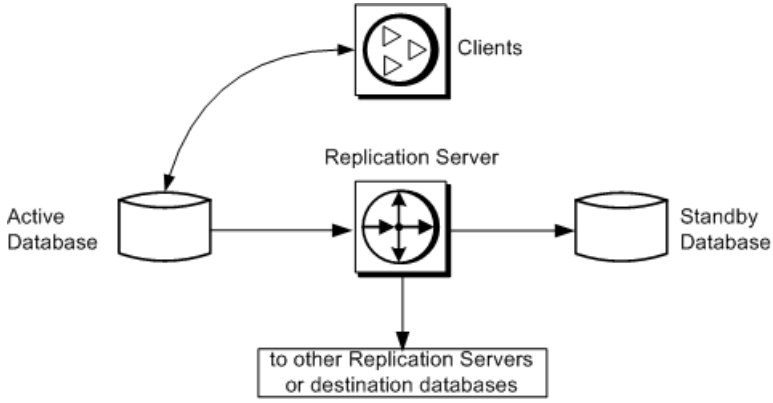
为使备用数据库与活动数据库保持一致，Replication Server 会重新生成从活动数据库的事务日志检索的事务信息。虽然复制定义便于向备用数据库复制数据，但它们不是必需的。将数据复制到备用数据库不需要预订。

热备份的工作原理

了解热备份的工作方式。

此图说明示例热备份应用程序的正常操作。

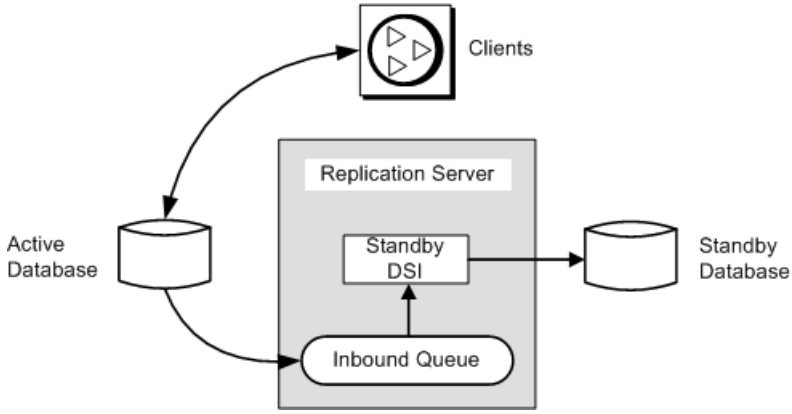
图 2： 热备份应用程序 - 正常操作



在此热备份应用程序中：

- 客户端应用程序在活动数据库中执行事务。
- 活动数据库的 RepAgent 从事务日志中检索事务并将它们转发到 Replication Server。
- Replication Server 在备用数据库中执行事务。
- Replication Server 也可以将事务复制到目标数据库和远程 Replication Server。

图 3： 热备份应用程序示例 – 切换之前



此图显示了有关热备份应用程序中的组件和进程的详细信息。

另请参见

- 切换活动数据库和备用数据库之前 （第 76 页）

热备份应用程序中的数据库连接

在热备份应用程序中，活动数据库和备用数据库在复制系统内的表现形式是从 Replication Server 到单个逻辑数据库的连接。

复制系统管理员通过创建此逻辑连接来为活动数据库和备用数据库建立一个符号名。

这样，一个热备份应用程序包括以下来自 Replication Server 的数据库连接：

- 活动数据库的物理连接
- 备用数据库的物理连接
- 活动数据库和备用数据库的逻辑连接

Replication Server 将逻辑连接映射到当前活动的数据库，并将事务从活动数据库复制到备用数据库。

请参见《Replication Server 管理指南第一卷》中的“管理数据库连接”。

为了提高复制性能，您可以在热备份环境中创建替代连接和替代逻辑连接。

另请参见

- 用于热备份环境的多个复制路径 （第 241 页）
- 设置 ASE 热备份数据库 （第 63 页）

主数据库和复制数据库以及热备份应用程序

逻辑数据库也可以充当主数据库或复制数据库。

在很多 Replication Server 应用程序中：

- 主数据库是数据的来源；复制过程使用复制定义和预订将这些数据复制到其它数据库。
- 复制数据库从主数据库中接收数据。

Replication Server 处理逻辑数据库的方式与处理所有其它数据库的方式相同。根据应用程序的不同，热备份应用程序中的逻辑数据库可以作为不参与复制的数据库而仅用于热备份，或者逻辑数据库也可以作为主数据库或复制数据库。

数据库关系比较
通常，将数据库定义为“主数据库”或“复制数据库”。然而，在讨论热备份应用程序时，还会将数据库定义为“活动数据库”或“备用数据库”。

表 6. 活动数据库和备用数据库与主数据库和目标数据库

活动数据库和备用数据库	主数据库和复制数据库
活动数据库和备用数据库必须由相同的 Replication Server 管理。	主数据库和目标数据库可以由相同的 Replication Server 管理，也可以由不同的 Replication Server 管理。
活动数据库和备用数据库必须是 Adaptive Server 数据库。	除非参与热备份应用程序，否则主数据库和目标数据库不必是 Adaptive Server 数据库。
活动数据库有一个备用数据库。 信息总是从活动数据库复制到备用数据库。	一个主数据库可以有一个或多个目标数据库。 某些数据库既包含主数据，又包含复制的数据。
复制定义的使用是可选的。不使用预订。	从主数据库复制到目标数据库时，必须使用复制定义和预订。
到备用数据库的连接使用函数字符串类 **rs_default_function_class**。 您无法为此类自定义函数字符串。	到复制数据库的连接可以使用函数字符串类，您可以在该类中自定义函数字符串。例如，该连接可能会使用从 **rs_default_function_class** 继承函数字符串的派生类。
可以切换活动数据库和备用数据库的角色。	不能切换主数据库和复制数据库的角色。
客户端应用程序通常连接到活动数据库。(不过，您可以对备用数据库执行只读操作。) 没有提供在将 Replication Server 切换到备用数据库时切换客户端应用程序的机制。	客户端应用程序可以连接到主数据库或目标数据库。只有主数据才能被直接修改。 一般情况下，客户端应用程序无需在主数据库与目标数据库之间切换。
对复制的表执行的所有事务（包括维护用户事务）均由活动数据库的 RepAgent 提交到 Replication Server，Replication Server 会在备用数据库中重新生成这些事务。 在目标数据库的热备份应用程序中，活动数据库中的事务通常由维护用户执行。	在大多数应用程序中，RepAgent 不将维护用户事务提交给 Replication Server 去在目标数据库中重新进行生成。 维护用户通常不执行主数据库中的事务。

另请参见

- 使用复制的热备份应用程序 （第 90 页）

热备份要求和限制

有一些要求和限制适用于所有 Replication Server 热备份应用程序。

- 您必须使用支持热备份应用程序的数据服务器，如 Adaptive Server。
- 一个 Replication Server 同时管理活动数据库和备用数据库。活动数据库和备用数据库都必须是 Adaptive Server 数据库。有关如何在两个 Oracle 数据库之间设置和

配置热备份应用程序的详细信息，请参见《Replication Server 异构指南》中的“Oracle 异构热备份”。

- 无法为 RSSD 创建备用数据库。只有在 Adaptive Server 支持主数据库复制（如 Adaptive Server 15.0 ESD #2 和更高版本）时，才能为 `master` 数据库创建备用数据库。
- Replication Server 不将客户端应用程序切换到备用数据库。
- 您应该在不同的计算机上为活动数据库和备用数据库运行 Adaptive Server。如果将活动数据库和备用数据库放在相同的数据服务器或硬件资源上，将会削弱热备份功能的优势。
- 虽然 Adaptive Server 允许包含重复行的表，但活动数据库和备用数据库中的表在每一行的主键列应该具有唯一的值。
- 抽象计划的各个命令和过程均会被复制，但下面的内容除外：
  - 不复制 **create plan** 的 **and set** *@plan_id* 子句。例如，该命令不按以下方式复制。

    ```
    create plan "select avg(price)
    from titles" "(t_scan titles)
    into dev_plans and set @plan_id
    ```

    而是按以下方式复制：

    ```
    create plan "select avg(price)
    from titles" "(t_scan titles)
    into dev_plans
    ```

  - 不复制将计划 ID 作为参数的抽象计划过程（**sp_drop_qplan**、**sp_copy_qplan**、**sp_set_qplan**）。
  - 不复制 **set plan** 命令。
- 故障切换支持不能替代热备份。虽然热备份会保留数据库的副本，但 Sybase 故障切换从不同的计算机访问同一个数据库。对于从 Replication Server 到热备份数据库的连接，故障切换支持的工作方式是相同的。

  请参见 Adaptive Server Enterprise 文档集中的《在高可用性系统中使用 Sybase 故障切换》。
- 无法在活动数据库中使用 dump and enable 标记并随后使用跨平台 **dump** 和 **load** 重建备用数据库。Replication Agent 必须将 dump 标记发送到要重建的备用数据库。在跨平台转储和装载期间，在从活动数据库中获取转储时，活动数据库必须处于单用户模式。

另请参见

- 设置客户端以使用活动数据服务器 （第 84 页）
- 配置复制系统以支持 Sybase 故障切换 （第 284 页）
- 跨平台转储和装载 （第 69 页）

用于维护备用数据库的函数字符串

Replication Server 将系统提供的函数字符串类 **rs_default_function_class** 用于备份 DSI，备份 DSI 是到备用数据库的连接。

Replication Server 会为该类生成缺省函数字符串。您不能自定义 **rs_default_function_class** 类中的函数字符串。

为热备份复制的信息

Replication Server 支持采用各种不同的方法向备用数据库复制数据。Replication Server 复制到备用数据库的信息的级别和类型取决于所选的方法。

您必须选择以下两种方法之一：

- 可以使用 **sp_reptostandby** 系统过程将整个数据库标记为要复制到备用数据库。**sp_reptostandby** 将启用对数据操纵语言 (DML) 命令和一组受支持的数据定义语言 (DDL) 命令以及系统过程的复制。
  - DML 命令（如 **insert**、**update**、**delete** 和 **truncate table**）用于更改用户表中的数据。
  - DDL 命令和系统过程用于更改数据库的模式或结构。

  sp_reptostandby 可用于复制对数据库中存储的系统表进行更改的 DDL 命令和过程。您可以使用 DDL 命令来创建、更改和删除数据库对象，例如，表和视图。受支持的 DDL 系统过程会影响有关数据库对象的信息。这些过程由初始用户在备用数据库中执行。
- 如果选择不使用 **sp_reptostandby**，则可以使用 **sp_setreptable** 将各个用户表标记为要复制。利用此过程，可以为标记的表复制 DML 操作。

另外，您也可以指示 Replication Server 将哪些用户存储过程复制到备用数据库：

- 通过使用 **sp_setrepproc** 系统过程标记用户存储过程，您可以将这些存储过程的执行复制到备用数据库。通常，只有与函数复制定义关联的存储过程才会复制到备用数据库。

有关为 Oracle 热备份复制的信息的详细信息，请参见《Replication Server 异构指南》中的“Oracle 异构热备份”。

另请参见

- 使用 sp_setrepproc 复制用户存储过程 （第 59 页）

复制方法的比较

比较 **sp_reptostandby** 和 **sp_setreptable** 以了解二者如何将信息复制到备用数据库。

表 7. 表复制方法的比较

sp_reptostandby	sp_setreptable
将所有用户表复制到备用数据库。	允许选择将哪些用户表复制到备用数据库。
允许复制 DML 命令和受支持的 DDL 命令及系统过程。	允许复制对所标记的表执行的 DML 命令。 **注意：** 您可以强制为 **isql** 会话复制支持的 DDL 操作。
不将 DML 和 DDL 操作复制到复制数据库。 如果热备份应用程序也将数据复制到复制数据库，则必须使用 **sp_setreptable** 标记要复制到复制数据库的表。	将 DML 操作复制到备用数据库和复制数据库。
将 **truncate table** 命令的执行复制到备用数据库。不需要预订。 **注意：** 您可以使用 **alter logical connection** 命令启用或禁用将 **truncate table** 复制到备用数据库的操作。	如果使用 Adaptive Server 数据库，则会将 **truncate table** 的执行复制到备用数据库。不需要预订。
Replication Server 使用表名和表所者信息来标识备用数据库中的表。	在标记要复制到热备份数据库的表时，如果包括 **owner_on** 关键字，Replication Server 将使用表名和表所有者信息来标识备用数据库中的表。 在标记要复制到热备份数据库的表时，如果包括 **owner_off** 关键字，则 Replication Server 将使用表名和“dbo”来标识备用数据库中的表。
缺省情况下，仅将更改的 `text`、`unitext`、`image` 和 `rawobject` 列复制到备用数据库。 如果使用 **sp_reptostandby** 和 **sp_setreptable** 标记数据库表，则可能以不同的方式对 `text`、`unitext`、`image` 和 `rawobject` 数据进行处理。	缺省情况下，`text`、`unitext` 和 `image` 列总是会复制到备用数据库。 如果用 **sp_setrepcol** 设置复制状态，则 `text`、`unitext`、`image` 和 `rawobject` 列将被视为带有以下标记：**always_replicate**、**replicate_if_changed** 或 **do_not_replicate**。
当活动数据库与备用数据库相同时，这是最简便的方法。	

另请参见

- 强制将 DDL 命令复制到备用数据库 （第 62 页）

- 将 truncate table 复制到备用数据库 （第 87 页）
- 在热备份应用程序中复制 text、unitext、image 和 rawobject 数据 （第 60 页）
- 支持的 DDL 命令和系统过程 （第 55 页）

使用 sp_reptostandby 启用复制

sp_reptostandby 用于将所有用户表的 DML 和支持的 DDL 命令复制到备用数据库。

要启用对 DML 和 DDL 命令的复制，请在管理活动数据库的 Adaptive Server 上执行 **sp_reptostandby**：

```
sp_reptostandby dbname, [[, 'L1' | 'ALL' | 'NONE' ] [, use_index]]
```

其中 *dbname* 是活动数据库名，关键字 **L1**、**all** 和 **none** 设置复制支持的级别。

L1 表示 Adaptive Server 12.5 版支持的复制级别。

使用 **all** 关键字可确保模式复制支持始终处于可用的最高级别。例如，要将模式复制支持级别设为 Adaptive Server 最新版本的级别，请登录到 Adaptive Server 并在 **isql** 提示符下执行以下命令：

```
sp_reptostandby dbname, 'all'
```

之后，如果数据库升级到具有更高复制支持级别的高版本 Adaptive Server，则会自动启用该版本的所有新功能。

如果 DDL 命令或系统过程包含口令信息，则将使用源 Adaptive Server 系统表中存储的密文口令值在复制环境下发送此口令信息。

请参见《Replication Server 参考手册》的“Adaptive Server 命令和系统过程”中的“**sp_reptostandby**”。

使用 sp_reptostandby 时的限制和要求

在设置热备份应用程序并使用 **sp_reptostandby** 启用复制时，应考虑这些限制和要求。

- 活动数据库和备用数据库必须由 Adaptive Server 管理，并且必须支持 RepAgent。两个数据库必须具有相同的磁盘分配、段名和角色。请参见《Adaptive Server Enterprise 系统管理指南》。
- 备份服务器中必须存在活动数据库名。否则，包含该数据库名的命令或过程的复制将失败。
- Replication Server 不支持复制包含局部变量的 DDL 命令。您必须为这些命令显式定义特定于节点的信息。
- 不会将登录信息复制到备用数据库。确保服务器用户的 ID 匹配，然后将登录信息添加到目标 Replication Server 中。
- 有些命令不复制到备用数据库：
  - **select into**
  - **update statistics**
  - 数据库或配置选项，例如 **sp_dboption** 和 **sp_configure**

- 在主 Adaptive Server 上执行 **set proxy** 后，Replication Server 不支持复制 DDL 命令；Replication Server 将返回错误 5517：

```
A REQUEST transaction to database '...' failed because the
transaction    owner's password is missing. This prevents the
preservation of transaction ownership.
```

另请参见

- 使服务器用户的 ID 匹配 （第 71 页）

支持的 DDL 命令和系统过程

使用 **sp_reptostandby** 启用复制时 Replication Server 在备用数据库上复制的 DDL 命令、Transact-SQL 命令和 Adaptive Server 系统过程。

有一部分命令和存储过程标有星号，这表示它们的复制在 Adaptive Server 12.5 及更高版本中受支持。

支持的 DDL 命令有：

- **alter encryption key**
- **alter key**
- **alter login**
- **alter login profile**
- **alter...modify owner** – Replication Server 将具有不同所有者的表视为不同表。如果您使用 **alter...modify owner** 更改 Adaptive Server 复制表的所有者，则必须对表复制定义进行相关更改。请参见《Replication Server 管理指南第一卷》的“管理复制表”的“修改复制定义”的“更改复制定义”的“可以对复制定义进行的更改”中的“更改表所有者”。
- **alter table**
- **create default**
- **create encryption key**
- **create function**
- **create index**
- **create key**
- **create login**
- **create login profile**
- **create plan***
- **create procedure**
- **create rule**
- **create schema***
- **create table**
- **create trigger**
- **create view**
- **drop default**

- **drop function**
- **drop login**
- **drop login profile**
- **drop index**
- **drop procedure**
- **drop rule**
- **drop table**
- **drop trigger**
- **drop view**
- **grant**
- **installjava*** – MSA 环境不支持复制 **installjava**。
- **remove java***
- **revoke**

受支持的系统过程有:

- **sp_add_qpgroup***
- **sp_addalias**
- **sp_addgroup**
- **sp_addmessage**
- **sp_addtype**
- **sp_adduser**
- **sp_bindefault**
- **sp_bindmsg**
- **sp_bindrule**
- **sp_cachestrategy**
- **sp_changegroup**
- **sp_chgattribute**
- **sp_commonkey**
- **sp_config_rep_agent**
- **sp_drop_all_qplans***
- **sp_drop_qpgroup***
- **sp_dropalias**
- **sp_dropgroup**
- **sp_dropkey**
- **sp_dropmessage**
- **sp_droptype**
- **sp_dropuser**
- **sp_encryption**
- **sp_export_qpgroup***

- **sp_foreignkey**
- **sp_hidetext**
- **sp_import_qpgroup***
- **sp_primarykey**
- **sp_procxmode**
- **sp_recompile**
- **sp_rename**
- **sp_rename_qpgroup***
- **sp_replication_path**
- **sp_setrepcol**
- **sp_setrepdefmode**
- **sp_setrepproc**
- **sp_setreplicate**
- **sp_setreptable**
- **sp_unbindefault**
- **sp_unbindmsg**
- **sp_unbindrule**

支持在主数据库中进行复制的 DDL 命令和系统过程集不同于支持在用户数据库中进行复制的 DDL 命令和系统过程集。

如果数据库是主数据库，则支持的 DDL 命令有：

- **alter role**
- **create role**
- **drop role**
- **grant role**
- **revoke role**

如果数据库是主数据库，则支持的系统过程为：

- **sp_addexternlogin**
- **sp_addlogin**
- **sp_addremotelogin**
- **sp_addserver**
- **sp_defaultdb**
- **sp_defaultlanguage**
- **sp_displaylevel**
- **sp_dropexternlogin**
- **sp_droplogin**
- **sp_dropremotelogin**
- **sp_dropserver**

- **sp_locklogin**
- **sp_maplogin**
- **sp_modifylogin**
- **sp_password**
- **sp_passwordpolicy** – 为 **allow password downgrade** 以外的所有选项复制。
- **sp_role**

复制 alter table：限制
当 Adaptive Server 执行 **alter table ... add** *column_name* **default** ... 语句时，服务器将使用 `objid` 为缺省值创建约束。

在 Replication Server 复制此语句之后，备用 Adaptive Server 将使用其它 `objid` 创建同一约束。

如果稍后使用 **alter table ... drop constraint ...** 在主数据库上删除此约束，则会由于 objid 不同而无法在热备份数据库上执行语句。

若要在主数据库和备用数据库上删除约束，请在主数据库中执行以下语句之一：

- ```
  alter table table_name
          ...
          replace column_name default null
  ```
- ```
  alter table table_name
          ...
          drop constraint constraint_name
  ```

  此语句将关闭 DSI。在备用数据库上将相同的命令与其相应的 `objid` 一起执行，然后恢复与 DSI 的连接，并跳过事务。

主数据库复制：限制
不会从主数据库中复制用户表和用户存储过程。

如果复制了主数据库，则必须在主数据库中执行以下系统过程：

- **sp_addlogin**
- **sp_defaultdb**
- **sp_defaultlanguage**
- **sp_displaylevel**
- **sp_droplogin**
- **sp_locklogin**
- **sp_modifylogin**

如果使用的数据库是主数据库，则源 Adaptive Server 和目标 Adaptive Server 都必须支持主数据库复制功能。

如果数据库是主数据库，则源 Adaptive Server 和目标 Adaptive Server 必须具有相同的硬件体系结构类型（32 位版本和 64 位版本可以兼容）和相同的操作系统（不同的版本也可以兼容）。

禁用复制

可以将 **sp_reptostandby** 与 **none** 选项一起使用以关闭数据和模式复制。

登录到 Adaptive Server，然后在 **isql** 提示符下输入：

```
sp_reptostandby dbname,'none'
```

关闭复制后，Adaptive Server 将在排它模式下锁定所有用户表，并保存每个用户表的相关信息。如果数据库中存在大量的用户表，则完成此过程可能需要一些时间。

只有 在禁用热备份应用程序时，才使用此过程。

注意： 要仅关闭当前 **isql** 会话的复制，请使用 **set replication** 命令。

另外，如果将数据库标记为要复制以使用 `text`、`unitext`、`image` 和 `rawobject` 列上的索引，则 **sp_reptostandby** *dbname,* **'none'** 也会删除未显式标记为要复制的表中要复制的索引。

另请参见

- 更改当前 isql 会话的复制 （第 62 页）

使用 sp_setreptable 启用复制

sp_setreptable 用于标记要复制到复制数据库或备用数据库的单个表。

Replication Server 将把对这些表执行的 DML 操作复制到备用数据库和复制数据库。

在以下情况下，请使用 **sp_setreptable** 标记要复制到备用数据库的表：

- 您使用 Adaptive Server 数据库，或者
- 您选择不使用 **sp_reptostandby**。

使用 **sp_setreptable** 可以维护在活动数据库和备用数据库之间的数据（但不是模式）一致性。**sp_setreptable** 正常情况下不会将受支持的 DDL 命令和过程复制到备用数据库。不过，您可以使用 **set replication** 命令强制复制当前 **isql** 会话的 DDL 命令。

如果数据库是主数据库，则不会复制用户表。

另请参见

- 更改当前 isql 会话的复制 （第 62 页）

使用 sp_setrepproc 复制用户存储过程

要将用户存储过程的执行复制到备用数据库，请用 **sp_setrepproc** 标记要复制的存储过程。

对于用 **sp_setrepproc** 标记的过程，如果已经为它们创建了预订，则还会在复制数据库中重新生成这些过程。

对于热备份应用程序中的存储过程的执行，有两种可能的情况：

- 如果已经用 **sp_setrepproc** 标记了要复制的存储过程，Replication Server 会将该过程的执行复制到备用数据库。它不会 将存储过程产生的效果复制到备用数据库。
- 如果尚未标记要复制的存储过程，那么，假如受影响的表已被标记为要复制，则 Replication Server 会将受该过程影响的 DML 更改复制到备用数据库。

有关 **sp_setrepproc** 系统过程的详细信息，请参见《Replication Server 管理指南第一卷》中的“管理复制函数”。

如果数据库是主数据库，则不会复制用户过程。

复制名称相同但所有者不同的表

Adaptive Server 和 Replication Server 允许复制名称相同但所有者不同的表。

在用 **sp_reptostandby** 标记要复制的数据库时，更新会被自动复制到备用数据库中具有相同名称和所有者的表。

在用 **sp_setreptable** 标记要复制的表时，可选择是否使用表所有者名称来选择备用数据库中正确的表。

- 如果设置 **owner_on**，Replication Server 便将表名和表所有者名称发送到备用数据库。
- 如果设置 **owner_off**，则 Replication Server 会将表名称和“dbo”作为所有者名称发送到备用数据库。

注意： 如果要将信息复制到复制数据库并且已使用 **sp_setreptable** 设置 **owner_off**，则 Replication Server 会将表名发送到复制数据库，但不发送所有者信息。

请参见《Replication Server 管理指南第一卷》的“管理复制表”的“将表标记为要复制”的“使用 sp_setreptable 系统过程”中的“使用 **owner_on** 状态启用复制”。

注意： 如果将一个具有非唯一的名称的表标记为要复制，并且为该表创建了复制定义，则必须在复制定义中包括所有者信息。否则，Replication Server 将无法在复制数据库或备用数据库中找到正确的表。

在热备份应用程序中复制 text、unitext、image 和 rawobject 数据

如果用 **sp_reptostandby** 标记数据库，则复制状态会自动设置为 **replicate_if_changed**，而 Adaptive Server 将只记录已更改的 `text`、`unitext`、`image` 和 `rawobject` 列。

这样可确保备用数据库与活动数据库保持同步。您不能使用 **sp_setrepcol** 更改这种表的复制状态。

如果用 **sp_setreptable** 标记要复制的表，则缺省的复制状态为 **always_replicate**，而 Adaptive Server 将记录所有的 `text`、`unitext`、`image` 和 `rawobject` 列数据。对于用 **sp_setreptable** 标记的表，您可以更改其中的 `text`、`unitext`、`image` 和 `rawobject` 列的复制状态。使用 **sp_setrepcol** 可将复制状态更改为 **replicate_if_changed** 或 **do_not_replicate**。一列或列组合必须唯一标识每一行。

如果使用复制定义，主键必须是一组可唯一地标识表中的每一行的列。确保 Adaptive Server 和 Replication Server 的复制状态相同。

使用复制数据库中的 use_index 选项

可以使用 **use_index** 选项加快对要复制的 `text`、`unitext`、`image` 或 `rawobject` 列的设置过程。

它对于包含一个或多个 `text`、`unitext`、`image` 或 `rawobject` 列的大表特别有用。可以将 **use_index** 选项设置为数据库级别、表级别或列级别。例如，可以对表进行标记而不使用索引，但可以只显式标记一个列以使用索引进行复制。

当您将 **use_index** 选项和 **sp_reptostandby** 一起使用时，数据库标记为使用 `text`、`unitext`、`image` 或 `rawobject` 列上的索引，并在未显式标记为要复制的表中创建内部索引。

对于标记为使用索引进行复制的数据库，如果创建带有行外列的新表，也会创建要复制的索引。类似地，当在标记为使用索引的数据库中执行 **alter table...add column** 命令时，将在行外列中创建内部索引。使用 **alter table...drop column** 命令时，如果要删除的列标记为使用索引，则也会删除用于复制的内部索引。

不同对象级别的复制索引状态的排列顺序为：列、表和数据库。如果将数据库标记为使用索引进行复制，但您在未使用索引的情况下对表进行了标记，则表状态将覆盖数据库状态。

注意： 行外（`text`、`unitext`、`image` 或 `rawobject`）列中的复制性能不会发生变化。只会影响对要复制的数据库、表或列进行标记的过程。

如果表包含大量的行或数据库包含一个或多个带有相当多的行和一些行外列的表，则可以使用 **use_index** 选项。

配置热备份数据库以进行 SQL 语句复制

缺省情况下，热备份应用程序不会复制支持 SQL 语句复制的 DML 命令。不过，可以通过几种方法使用 SQL 语句复制。

- 使用 **replicate SQLDML** 和 **send standby** 子句创建表复制定义。
- 将 **ws_sqldml_replication** 参数设置为 on。缺省值为 **UDIS**。不过，**ws_sqldml_replication** 的优先级比 SQL 复制的表复制定义低。如果表的表复制定义包含 **send standby** 子句，该子句将确定是否复制 DML 语句，而无论 **ws_sqldml_replication** 参数设置是什么。

复制加密列

在热备份应用程序中使用加密列时的注意事项与非热备份环境类似。

请参见《Replication Server 管理指南第一卷》的“管理复制表”中的“复制加密列”。

复制带引号的标识符

如果复制到热备份数据库和复制定义预订者，并且将主表名称标记为带引号的标识符但未标记复制表名称（或相反），则 Replication Server 将主表名称和复制表名称均作为带引号的标识符发送。

热备份包含复制数据库时

您可以将活动数据库中的信息复制到备用数据库，同时还可以将这些信息复制到复制数据库。

Replication Server 必须将表的 `text`、`unitext`、`image` 和 `rawobject` 列复制到具有相同复制状态的备用和复制数据库。

如果要将所有 `text`、 `unitext`、`image` 和 `rawobject` 列复制到备用数据库和复制数据库，则不要更改表的复制状态。缺省情况下，所有的 `text`、 `unitext`、`image` 和 `rawobject` 列都会被复制到备用数据库和复制数据库。

要仅复制已更改的`text`、`unitext`、`image`和`rawobject`列，请使用 **sp_setrepcol** 将复制状态设为 **replicate_if_changed**。

更改当前 isql 会话的复制

您可以使用 **set replication** 控制 **isql** 会话的 DML 和 DDL 命令及过程的复制。

在管理活动数据库的 Adaptive Server 上执行 **set replication**。语法为：

```
set replication [on | force_ddl | default | off]
```

缺省设置为“on”。缺省行为取决于是否已使用 **sp_reptostandby** 将数据库标记为要进行复制。

表 8. set replication 的缺省行为

如果已使用 sp_reptostandby 将数据库标记为要复制	如果尚未使用 sp_reptostandby 将数据库标记为要复制
Replication Server 将所有用户表的 DML 和支持的 DDL 命令复制到备用数据库。	Replication Server 只将 **sp_setreptable** 标记过的表的 DML 命令复制到备用数据库和复制数据库

请参见《Replication Server 参考手册》的“Adaptive Server 命令和系统过程”中的“**set replication**”。

强制将 DDL 命令复制到备用数据库

可以使用 **set replication force** 强制复制支持的 DDL 命令和系统过程。

例如，要为 **isql** 会话强制复制所有支持的 DDL 命令和系统过程，请输入：

```
set replication force_ddl
```

此命令为 **sp_setreptable** 标记的表启用 DDL 命令和系统过程复制。

要关闭 **force_ddl** 并将 **set replication** 恢复缺省状态，请输入：

```
set replication default
```

关闭所有向备用数据库执行的复制操作

可以使用 **set replication force off** 关闭所有向备用数据库执行的复制操作。

要为 **isql** 会话关闭所有向备用数据库执行的复制操作，请输入：

```
set replication off
```

设置 ASE 热备份数据库

为热备份应用程序设置数据库包括一些高级任务。

1. 创建同时供活动数据库和备用数据库使用的单个逻辑连接。
2. 使用 Sybase Central 或 **rs_init** 将活动数据库添加到复制系统。
   如果您已经将一个以前添加到复制系统中的数据库指定为活动数据库，则不需要添加活动数据库。
3. 使用 **sp_reptostandby** 或 **sp_setreptable** 为活动数据库中的表启用复制。
4. 使用 Sybase Central 或 **rs_init** 将备用数据库添加到复制系统，然后初始化备用数据库。

开始之前

设置 ASE 热备份数据库时，应满足一些前提条件。

- 必须安装并运行管理活动数据库和备用数据库的 Replication Server。用一个 Replication Server 同时管理活动数据库和备用数据库。
- 必须安装并运行包含活动数据库和备用数据库的 Adaptive Server。理论上，这些数据库应该由不同计算机上运行的数据服务器管理。
- 将数据库作为活动数据库或备用数据库添加到复制系统之前，该数据库必须已存在于 Adaptive Server 中。

另请参见

- 热备份要求和限制 （第 50 页）

客户端应用程序问题

在设置热备份数据库之前，应考虑一些客户端应用程序问题。

根据客户端应用程序和备用数据库初始化方法的不同，您可以挂起活动数据库中的事务处理，直到完成备用数据库的初始化。

如果您不挂起事务处理，则应确保 Replication Server 有足够的稳定队列空间，以存放在您将数据装载到备用数据库中时执行的事务。

在设置热备份数据库之前，应实现将客户端应用程序切换到新活动数据库的机制。

另请参见

- 设置客户端以使用活动数据服务器 （第 84 页）

任务 1：创建逻辑连接

如果活动数据库已是复制系统的一部分，请为活动数据库创建逻辑连接并重新配置 RepAgent。

另请参见

- 热备份应用程序中的数据库连接 （第 49 页）

命名逻辑连接

为逻辑连接指定的名称取决于是否将活动数据库添加到复制系统中。

在创建逻辑连接时，请将逻辑数据服务器名和逻辑数据库名组合在一起，形式为 *data_server.database*，并且：

- 如果活动数据库尚未添加到复制系统中 – 对逻辑连接和活动数据库分别使用不同的名称。对逻辑连接和物理连接使用唯一的名称可以使活动数据库的切换更加直接。
- 如果以前已将活动数据库添加到复制系统中 – 使用活动数据库的 *data_server* 和 *database* 名称作为逻辑连接名称。逻辑连接继承引用此物理数据库的所有现有复制定义和预订。

在为热备份应用程序创建复制定义或预订时，应指定逻辑连接，而不指定物理连接。如果指定逻辑连接，Replication Server 即可引用当前的活动数据库。

另请参见

- 使用复制的热备份应用程序 （第 90 页）

创建逻辑连接的过程

可以使用 **create logical connection** 命令从 Replication Server 中创建连接。

1. 登录到将要管理热备份数据库的 Replication Server，所用的登录名应具备 **sa** 权限。
2. 执行 **create logical connection** 命令：

   ```
   create logical connection to data_server.database
   ```

   数据服务器名可以是任何有效的 Adaptive Server 名，数据库名可以是任何有效的数据库名。

重新配置和重新启动 RepAgent

在创建逻辑连接后，重新配置并重新启动 RepAgent。

如果将以前添加到复制系统中的某个数据库指定为活动数据库，那么在创建逻辑连接时，活动数据库的 RepAgent 线程将关闭。

1. 使用 **sp_config_rep_agent** 重新配置 RepAgent 以设置 **send_warm_standby_xacts** 配置参数。

   请参见《Replication Server 管理指南第一卷》的“管理 RepAgent 和支持 Adaptive Server”中的“设置 RepAgent”以及《Replication Server 参考手册》的“Adaptive Server 命令和系统过程”中的“**sp_config_rep_agent**”。
2. 重新启动 RepAgent。

任务 2：添加活动数据库

可以使用 **rs_init** 将数据库作为热备份应用程序的活动数据库添加到复制系统中。

按照适用于您的平台的 Replication Server 安装和配置指南中的说明，执行将数据库添加到复制系统中的步骤。

任务 3：为活动数据库中的对象启用复制

可以使用 **sp_reptostandby** 为存储过程启用复制，以及使用 **sp_reptostandby** 或 **sp_setreptable** 为活动数据库中的表启用复制。

您可以使用以下任一方法为活动数据库中的表启用复制：

- 使用 **sp_reptostandby** 标记要复制的数据库，从而启用对数据更改和所支持模式更改的复制。
- 使用 **sp_setreptable** 分别标记要复制数据更改的表。

1. 作为系统管理员或数据库所有者登录到 Adaptive Server，并执行：

   ```
   use active_database
   ```
2. 使用以下三种方法之一标记要复制的数据库表。

   - 通过执行 **sp_reptostandby** 系统过程来标记所有用户表：

     ```
     sp_reptostandby dbname, [ 'L1' | 'all' ]
     ```

     其中 *dbname* 是活动数据库的名称，**L1** 将复制级别设置为 Adaptive Server 11.5 版的复制级别，**all** 将复制级别设为 Adaptive Server 当前版本的复制级别。此方法同时复制 DML 和 DDL 命令及过程。
   - 通过使用 **use_index** 选项执行 **sp_reptostandby**，标记所有用户表：

     ```
     sp_reptostandby dbname, [[, 'L1' | 'ALL'][, use_index]]
     ```

其中，*dbname* 是活动数据库的名称。通过使用 **use_index** 选项，可将数据库标记为使用 text、unitext、image 或 rawobject 列上的索引，并在未显式标记为要复制的那些表中创建内部索引。

- 通过对要复制到备用数据库中的每个表执行 **sp_setreptable** 系统过程，标记各个要复制数据更改的表：

  ```
  sp_setreptable table_name, 'true'
  ```

  其中 *table_name* 是表名。此方法复制 DML 命令。

3. 使用相关参数为具有要复制到备用数据库的执行的每个存储过程执行 **sp_setrepproc**。

   - 如果使用《Replication Server 管理指南第一卷》的“管理复制函数”中介绍的复制函数，则使用 **'function'** 参数执行 **sp_setrepproc**：

     ```
     sp_setrepproc proc_name, 'function'
     ```

   - 如果使用异步过程（如与表复制定义关联的复制存储过程），则使用 **'table'** 参数执行 **sp_setrepproc**：

     ```
     sp_setrepproc proc_name, 'function'
     ```

另请参见

- 为热备份复制的信息 （第 52 页）
- 异步过程 （第 331 页）

为以后添加的对象启用复制

标记和添加要复制到备用数据库中的新表和用户存储过程。

- 如果用 **sp_reptostandby** 标记要复制的数据库，新表会自动标记为要复制。
- 如果用 **sp_setreplicate** 标记要复制到备用数据库的数据库表，则必须使用 **sp_setreplicate** 标记要复制的每个新表。
- 您必须使用 **sp_setrepproc** 标记要复制的每个新用户存储过程。

任务 4：添加备用数据库

请使用 **rs_init** 将备用数据库及其 RepAgent 添加到复制系统，然后用活动数据库的数据初始化备用数据库。

在将备用数据库添加到复制系统后，您必须做好操作准备。

然后，您可以为备用数据库中的对象启用复制以及为备用数据库中的维护用户授予权限。是否需要执行这些步骤取决于您用来初始化备用数据库的方法。

1. 如果备用数据库不存在，则创建该数据库。
2. 确定如何初始化备用数据库。
3. 如果使用 **dump** 和 **load** 初始化备用数据库，则添加备用数据库维护用户。
4. 在复制之前使用 **online database** 子句使新的数据库联机。

创建备用数据库

如果尚不存在备用数据库，必须根据需要在相应的 Adaptive Server 中创建该数据库。

有关创建数据库的详细信息，请参见《Adaptive Server Enterprise 系统管理指南》。

确定如何初始化备用数据库。

使用活动数据库的数据来初始化备用数据库。

可以使用以下 Adaptive Server 命令和实用程序来初始化备用数据库：

- **dump** 和 **load**，或者
- **bcp**，或者
- **quiesce database ... to** *manifest_file* 用于生成清单文件；**mount** 用于将数据复制到备用数据库。

请参见《Adaptive Server Enterprise 参考手册：命令》。

当您使用 Sybase Central 或 **rs_init** 添加备用数据库时，Replication Server 将在活动数据库事务日志中写入“启用复制”标记。当您执行转储数据库或转储事务时，Adaptive Server 会将转储标记写入活动数据库事务日志中。

如果在初始化过程中不挂起事务处理：

- 选择 Sybase Central 中的“dump marker”选项或选择 **rs_init**，并使用 **dump** 和 **load** 命令。

如果在初始化过程中挂起事务处理：

- 不要使用 Sybase Central 中的“dump marker”选项或 **rs_init**，并使用 **dump** 和 **load** 命令，或者
- 使用 **bcp**，或者
- 使用 **quiesce database ... to** *manifest_file* 和 **mount**。

如果使用的数据库是主数据库，则无法使用 **dump** 或 **load** 实现目标数据库。您可以使用其它方法，例如使用 **bcp** 就可以对数据进行处理以解决不一致问题。

数据库初始化方法摘要

应考虑这些标记的每个初始化方法和角色的问题。

表 9. 初始化备用数据库中的问题

问题	带“dump marker”使用 dump 和 load	不带“dump marker”使用 dump 和 load	使用 bcp	使用 mount
处理客户端应用程序。	如果不能挂起客户端应用程序的事务处理，则使用此方法。	如果可以挂起客户端应用程序的事务处理，则使用此方法。		如果可以挂起客户端应用程序的事务处理，则使用此方法。
Replication Server 何时开始向备用数据库复制数据？	Replication Server 从启用复制标记后的第一个转储标记开始向备用数据库复制数据。	Replication Server 从启用复制标记后开始向备用数据库复制数据。		Replication Server 从启用复制标记后开始向备用数据库复制数据。
创建维护用户登录名并确保所有用户 ID 匹配。	在活动 Adaptive Server 和备份 Adaptive Server 中添加备用数据库维护用户的登录名，并确保服务器用户的 ID 匹配。 （因为如果使用 **dump** 和 **load** 用活动数据库的数据初始化备用数据库，备用数据库中所有以前的内容都会替换为活动数据库的内容，所以应在活动 Adaptive Server 中创建登录名。）		当您添加备用数据库时，Sybase Central 或 **rs_init** 会在备份 Adaptive Server 和备用数据库中添加维护用户登录名和用户。	在活动 Adaptive Server 和备份 Adaptive Server 中添加备用数据库维护用户的登录名。确保服务器用户 ID 匹配。（因为如果使用 **mount** 用活动数据库的数据初始化备用数据库，备用数据库中所有以前的内容都会替换为活动数据库的内容，所以应在活动 Adaptive Server 中创建登录名。）
初始化备用数据库。	使用 **dump** 和 **load** 将数据从活动数据库传送到备用数据库。 您可以使用数据库转储和/或事务转储。		使用 **bcp** 将每个复制的表从活动数据库复制到备用数据库。	使用 **quiesce database ... to** *manifest_file* 和 **mount database** 将数据从活动数据库传送到备用数据库。
活动数据库连接状态。	与活动数据库的连接不会更改。	Replication Server 挂起与活动数据库的连接。		Replication Server 挂起与活动数据库的连接。

问题	带“dump marker”使用 dump 和 load	不带“dump marker”使用 dump 和 load	使用 bcp	使用 mount
恢复连接。	恢复与备用数据库的连接。	恢复与活动数据库和备用数据库的连接；恢复活动数据库中的事务处理。		恢复与活动数据库和备用数据库的连接；恢复活动数据库中的事务处理。

跨平台转储和装载

可以使用跨平台 dump 和 load 初始化具有 RepAgent 的备用数据库。

1. 在活动数据库上：
   a) 使用 **sp_stop_rep_agent** *database* 停止 RepAgent。
   b) 使用 **dbcc settrunc(‘ltm’, ‘ignore’)** 删除辅助截断点。
   c) 在 Adaptive Server 中，在单用户模式下设置数据库。

      输入：

      ```
      sp_dboption database_name, ‘single user’, true
      ```
   d) 对数据库执行检查点操作。

      输入：

      ```
      checkpoint
      ```
   e) 通过在 Adaptive Server 执行以下命令，转储数据库事务日志：

      ```
      dump tran database_name with truncate_only
      go
      ```
   f) 获取数据库的转储。
2. 在备用数据库上：
   a) 装载从备用数据库中获取的转储。

      Sybase 建议您运行 **sp_post_xpload** 以检查并重建索引，即使平台的端类型是相同的。
   b) 转储事务日志以删除 **sp_post_xpload** 创建的日志记录：

      ```
      dump tran database_name with truncate_only
      go
      ```
   c) 执行 Adaptive Server **sp_indsuspect** 系统过程以检查用户表中是否有标记为可疑的索引。
   d) 如果需要，请重建可疑索引。如果字符集或排序顺序发生变化，您必须执行 **sp_indsuspect** 并再次重建索引，直到 sp_indsuspect 不显示任何具有可疑索引的表为止。
   e) 执行 **dbcc settrunc ('ltm', 'valid')** 以恢复数据库日志中的辅助截断点，然后执行 **rs_zeroltm** 以将数据库定位符值重置为零。

      通过执行这些命令，RepAgent 可以在辅助截断点处开始。
   f) 使用 **sp_start_rep_agent** *database* 启动 RepAgent。

请参见《Adaptive Server Enterprise 系统管理指南第二卷》的第 11 章“制订备份和恢复计划”中的“跨平台转储和装载数据库”。

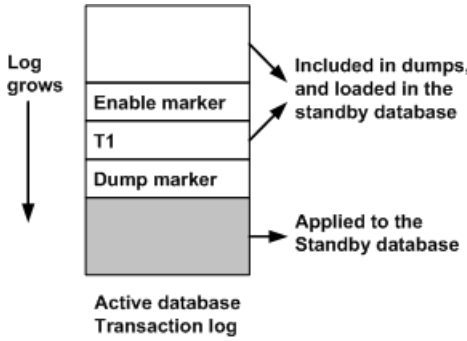
如果不挂起事务处理
如果在初始化备用数据库时不挂起活动数据库的事务处理，则在添加备用数据库时，请选择“dump marker”选项。

然后使用 **dump** 和 **load** 命令初始化备用数据库。

Replication Server 从活动数据库事务日志中的启用复制标记后的第一个转储标记开始向备用数据库复制数据。

在该图中，在添加备用数据库之后执行的事务 T1 在日志中出现于启用复制标记之后。T1 包含在转储中，因此在装载转储之后，T1 会出现在备用数据库中。Replication Server 不需要将它复制到备用数据库中。

图 4：使用带有转储标记的 dump 和 load 命令



在写入启用复制标记与转储活动数据库中的数据之间的这段时间中，可以在活动数据库中执行事务。

您可以将最后一个完整数据库转储及任何后续事务转储装载到备用数据库中，直到两个标记都已收到且备用数据库的操作准备就绪。之后，您可以选择使用活动数据库的最终事务转储来更新备用数据库。未包含在转储中的所有事务都会被复制。

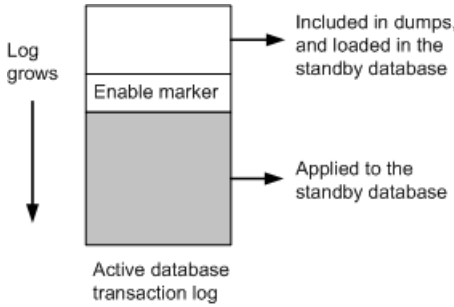
Replication Server 在收到启用复制标记和随后的第一个转储标记后，才会将事务从活动数据库复制到备用数据库。在收到这两个标记后，Replication Server 立即开始执行备用数据库中的事务。

如果挂起事务处理
如果在初始化备用数据库时挂起活动数据库的事务处理，则在添加备用数据库时不要选择“转储标记”选项。

您可以使用 **dump** 和 **load** 命令、**bcp** 或 **mount** 命令初始化备用数据库。

Replication Server 从活动数据库事务日志中的启用复制标记后开始向备用数据库复制数据。因为客户端应用程序已被挂起，所以启用复制标记后不会出现任何事务。

图 5：使用不带转储标记的 dump 和 load 命令或使用 bcp



如图所示，从写入启用复制标记时到使用 **dump** 命令转储（或使用 **bcp** 或 **mount** 复制）活动数据库中的数据时，没有在活动数据库中执行事务。

您可以将最后一个完整数据库转储或用 **bcp** 复制的最后一组复制表装载到备用数据库中，直到备用数据库收到启用复制标记。

收到此标记后，Replication Server 立即开始执行备用数据库中的事务。

添加备用数据库维护用户

如果计划使用 **dump** 和 **load** 命令在使用或不使用“dump marker”选项的情况下初始化备用数据库，则必须同时在备用和活动数据服务器中为备用数据库创建维护用户登录名，然后再添加备用数据库。

当您添加活动数据库时，Sybase Central 和 **rs_init** 都会自动在活动数据服务器中添加活动数据库维护用户。

使服务器用户的 ID 匹配

在每个数据服务器内，在 `master` 数据库中的 `syslogins` 表和每个用户数据库中的 `sysusers` 表中每个登录名的服务器用户 ID (`suid`) 必须相同。

此规则同样适用于热备份应用程序中的活动数据库和备用数据库。在 `master` 数据库中，`syslogins` 和 `sysloginroles` 表中的服务器用户的 ID 和角色设置也必须是相同的。

使用下面三种方法之一以使服务器用户的 ID 匹配：

- 将所有登录名（包括维护用户名）以相同的顺序添加到两个 Adaptive Server 中。Adaptive Server 按顺序分配服务器用户 ID，因此所有登录名的服务器用户 ID 都会匹配。
- 在将转储装载到备用数据库后，对备用数据库中的 `sysusers` 表和备份 Adaptive Server 的 `master` 数据库中的 `syslogins` 表进行调和。

- 维护一个主 Adaptive Server，在其中存放所有登录名，并将 `syslogins` 表从主 Adaptive Server 的 `master` 数据库复制到所有新创建的 Adaptive Server。

添加维护用户
将备用数据库的维护用户登录名添加到备份数据服务器和活动数据服务器中。

1. 在备份数据服务器中执行 **sp_addlogin** 系统过程，以创建维护用户登录名。

   有关使用 **sp_addlogin** 的详细信息，请参见《Adaptive Server Enterprise 系统管理指南》。
2. 在活动数据服务器中执行 **sp_addlogin**，以创建与在备份数据服务器中创建的维护用户登录名相同的登录名。

   当您用 **dump** 和 **load** 命令设置备用数据库时，`sysusers` 表将与其它数据一起从活动数据库装载到备用数据库中。

将备用数据库添加到复制系统中
初始化备用数据库，使其联机，然后恢复到该数据库的连接以将其添加到复制系统中。

1. 如果客户端应用程序和备用数据库的初始化方法允许挂起活动数据库中的事务处理，则挂起这些事务处理。

   如果不挂起事务处理，则必须带“dump marker”方法使用 **dump** 和 **load**。
2. 使用 Sybase Central 或 **rs_init** 将备用数据库添加到复制系统。执行文中所述的将数据库添加到复制系统的步骤。
3. 随时监控逻辑连接的状态。

   输入：

   ```
   admin logical_status, logical_ds, logical_db
   ```

   `Operation in Progress` 和 `State of Operation in Progress` 输出列指明备用数据库的创建状态。
4. 如果使用 **dump** 和 **load** 初始化备用数据库，则使用 **dump** 命令转储活动数据库的内容并装载备用数据库。

   例如：

   ```
   dump database active_database to dump_device
   ```

   ```
   load database standby_database from dump_device
   ```
5. 如果已装载了以前的数据库转储及后续的事务转储，则可以只转储事务日志并将它装载到备用数据库中。

   例如：

   ```
   dump transaction active_database to dump_device
   ```

   ```
   load transaction standby_database from dump_device
   ```

6. 完成装载操作后，使备用数据库联机：

   ```
   online database standby_database
   ```

   有关使用 **dump** 和 **load** 命令及 **online database** 命令的帮助信息，请参见《Adaptive Server Enterprise 参考手册》。

7. 初始化备用数据库。使用 **bcp** 或 **quiesce ... to***manifest_file* 和 **mount**。

   - 要使用 **bcp** 初始化备用数据库，请将复制的每个表从活动数据库复制到备用数据库。

     您必须复制 `rs_lastcommit` 表，该表是您将活动数据库添加到复制系统时创建的。

     有关使用 **bcp** 程序的帮助信息，请参见 Adaptive Server 实用程序手册。
   - 要使用 **quiesce ... to** *manifest_file* 和 **mount** 初始化备用数据库，应停顿数据库并创建清单文件。创建数据库和日志设备的副本。在备用数据库上装入设备。

8. 如果使用不带“dump marker”方法的 **dump** 和 **load** 初始化备用数据库、使用 **bcp** 初始化备用数据库或使用 **quiesce database ... to***manifest_file* 和 **mount** 初始化备用数据库，Replication Server 将挂起与活动数据库的连接。您必须恢复到活动数据库的连接。

   在 Replication Server 上，输入：

   ```
   resume connection to active_ds.active_db
   ```

9. 无论使用哪种初始化备用数据库的方法，必须恢复与备用数据库的连接。

   在 Replication Server 上，输入：

   ```
   resume connection to standby_ds.standby_db
   ```

10. 恢复活动数据库中的事务处理（如果它被挂起）。

在创建备用数据库时使用阻塞命令

可以使用 **wait for create standby** Replication Server 阻塞命令指示 Replication Server 不接受命令，直到备用数据库做好操作准备。

您可以在创建备用数据库的脚本中使用此命令。语法为：

```
wait for create standby for logical_ds.logical_db
```

为备用数据库中的对象启用复制

为了作好切换到备用数据库的准备，对于原备用数据库中要在切换后复制到新备用数据库的表和存储过程，必须启用复制。

- 如果使用 **dump** 和 **load** 或 **mount** 命令初始化备用数据库，备用数据库中的表和存储过程的复制设置将与活动数据库中的复制设置相同。
- 如果使用 **bcp** 初始化备用数据库，则使用 **sp_setreptable** 或 **sp_reptostandby** 和 **sp_setrepproc** 为这些对象启用复制。要为备用数据库中的对象启用复制，请调整为活动数据库中的对象启用复制的过程。

另请参见

- 任务 3：为活动数据库中的对象启用复制 （第 65 页）

为以后添加的对象启用复制

以后，您可以添加要复制到新备用数据库中的新表和新用户存储过程。

- 如果用 **sp_reptostandby** 标记要复制的备用数据库，所有新表也会自动标记为要复制。
- 如果用 **sp_setreplicate** 标记要复制到新备用数据库的单个数据库表，则必须使用 **sp_setreplicate** 标记要复制的每个新表。
- 您必须使用 **sp_setrepproc** 标记要复制的每个新用户存储过程。

向维护用户授予权限

在添加备用数据库后，必须向维护用户授予必要的权限。

1. 以系统管理员或数据库所有者身份登录到 Adaptive Server，并指定要使用的数据库。

   输入：

   ```
   use standby_database
   ```

2. 为维护用户授予 **replication_role**。

   输入：

   ```
   sp_role "grant", replication_role, maintenance_user
   ```

   replication_role 确保维护用户可以在备用数据库中执行 **truncate table**。

3. 为每个表执行 **grant all** 命令。

   输入：

   ```
   grant all on table_name to maintenance_user
   ```

在 ASE 热备份环境中复制主数据库

在 Adaptive Server 热备份环境中复制主数据库时，应满足一些要求和限制。

可以将 Adaptive Server 登录信息从一个主数据库复制到另一个主数据库。主数据库复制仅限于 DDL，并且使用系统命令来管理登录名和角色。主数据库复制不从系统表复制数据，也不从主数据库中的任何其它用户表复制数据或过程。

源 Adaptive Server 和目标 Adaptive Server 都必须是相同硬件体系结构类型（32 位版本与 64 位版本兼容）和相同操作系统（不同版本也兼容）。

不要用 **load** 从另一个主数据库初始化活动数据库和备用数据库。若要同步每个主数据库上的 **syslogins**、**suids** 和角色，请在设置复制之前使用 **bcp** 刷新相应表或手动同步 ID 和角色。

设置热备份应用程序和使用 **sp_reptostandby** 启用复制存在一些限制和要求，并且具有一些适用于主数据库的受支持的 DDL 和系统过程。

Sybase 建议您在备用主数据库上设置比活动主数据库更长的口令有效期。这样，活动主数据库就可以控制对口令的任何更改，并完成复制口令更改的操作。

Replication Server 12.0 和更高版本支持在热备份环境中的主数据库复制，在 Replication Server 12.6 和更高版本中还支持在 MSA 环境中进行主数据库复制。主或活动 Adaptive Server 的版本必须是 15.0 ESD #2 或更高版本。

有关 MSA 环境中的主数据库复制的信息，请参见《Replication Server 管理指南第一卷》的“使用多节点可用性管理复制对象”中的“在 MSA 环境中复制主数据库”。

另请参见

- 使用 sp_reptostandby 时的限制和要求 （第 54 页）
- 支持的 DDL 命令和系统过程 （第 55 页）

在热备份环境中设置主数据库复制

在热备份环境中设置主数据库复制。

1. 在 Replication Server 中将活动主数据库和备用主数据库设置为热备份对。

   不要“用 dump 和 load 初始化备用数据库”，也不要“使用 dump marker 启动到备用数据库的复制”。若要同步所有主数据库上的 **syslogins** 和 **suids**，请使用 **bcp**，或手动同步 ID。
2. 在活动和备用数据库上标记主数据库以发送系统过程。

   输入：

   ```
   sp_reptostandby master, ‘all’
   ```
3. 停止活动主数据库上的 RepAgent。

   输入：

   ```
   sp_stop_rep_agent master
   ```
4. 在活动和备用数据库上都配置复制代理以发送热备份事务。

   输入：

   ```
   sp_config_rep_agent master, ‘send warm standby
   xacts’, ‘true’
   ```
5. 重新启动活动主数据库上的 RepAgent。

   输入：

   ```
   sp_start_rep_agent master
   ```
6. 恢复与 Replication Server 上的活动和备用主数据库的 DSI 连接。

   输入：

   ```
   resume connection to active_ds.master
   go
   resume connection to standby_ds.master
   go
   ```
7. 验证热备份的状态。

   输入：

```
admin logical_status
```

另请参见

- 设置 ASE 热备份数据库 （第 63 页）

切换活动数据库和备用 ASE 数据库

如果活动数据库发生故障或希望在活动数据库上执行维护操作，您可以切换到备用数据库。

确定是否需要切换

您应该根据应用程序的要求确定何时需要从活动数据库切换到备用数据库。

通常，活动数据服务器发生瞬时故障时，不应进行切换。Adaptive Server 要从瞬时故障中恢复，只要重新启动即可，无需执行其它恢复步骤。如果活动数据库将会长时间不可用，您可能应该进行切换。

确定何时切换的因素包括：活动数据库需要的恢复程度、活动数据库与备用数据库的同步程度、用户或应用程序能够容许的停机时间长度。

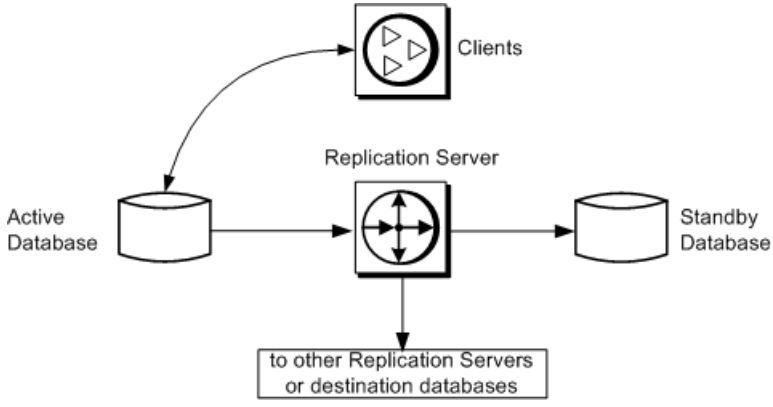
您也可能想切换活动数据库和备用数据库的角色，以便对活动数据库或其数据服务器执行事先计划的维护任务。

切换活动数据库和备用数据库之前

了解所涉及的过程以及在从活动数据库切换到备用数据库前热备份环境中组件的状态。

此图说明示例热备份应用程序的正常操作。

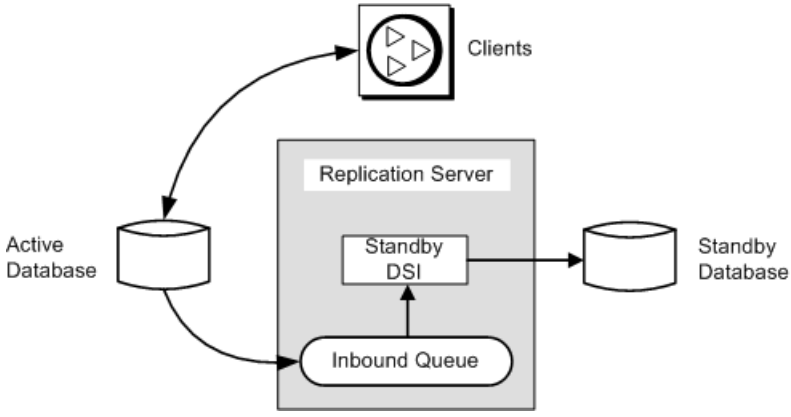
图 6： 热备份应用程序



“热备份应用程序示例 – 切换之前”图：

- 说明不参与复制系统（除热备份应用程序自身的活动之外）的数据库的热备份应用程序。
- 表示在切换活动和备用数据库之前正常操作中的热备份应用程序。
- 添加了内部细节以显示：
  - Replication Server 将从活动数据库收到的事务写入入站消息队列。
    有关入站和出站队列的详细信息，请参见《Replication Server 管理指南第一卷》的“Replication Server 技术概述”的“使用 Replication Server 处理事务”中的“分布式并发控制”。
  - 此入站队列由在备用数据库中执行事务的备用数据库的 DSI 线程读取。
    从活动数据库接收的消息要等到备用数据库的 DSI 线程读取了这些消息并将它们应用于备用数据库之后才能从入站队列中被截断。

图 7： 热备份应用程序示例 – 切换之前



在本示例中，事务只是从活动数据库复制到备用数据库。逻辑数据库本身并不：

- 包含复制到复制数据库或远程 Replication Server 的主数据，或者
- 从其它 Replication Server 接收复制的事务

另请参见

- 使用复制的热备份应用程序 （第 90 页）

内部切换步骤

当您切换活动数据库和备用数据库时，Replication Server 将执行一些任务。

Replication Server：

1. 对活动和备用 RepAgent 连接发出日志挂起。
2. 读取入站队列中所有剩余的消息并将它们应用于备用数据库，如果有预订数据或复制的存储过程，则应用于出站队列。

入站队列中所有提交的事务必须在切换结束之前处理。

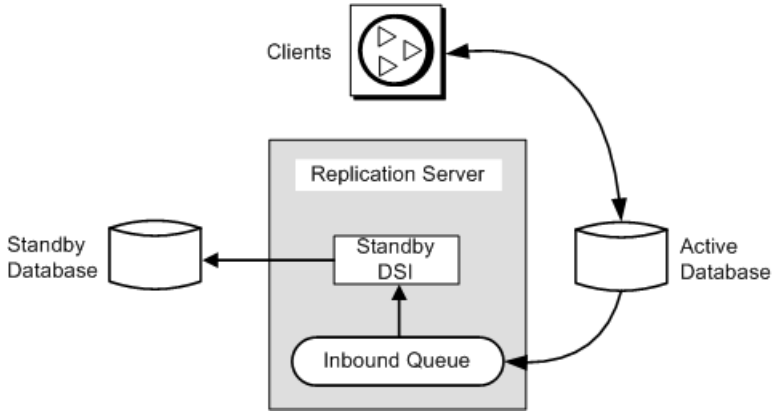
3. 挂起备份 DSI。
4. 启用新活动数据库中的辅助截断点。
5. 在新活动数据库的事务日志中放置标记。Replication Server 使用此标记确定将哪些事务应用于新的备用数据库，哪些事务应用于所有复制数据库。
6. 更新热备份数据库的 RSSD 中的数据。
7. 恢复新活动数据库的连接，并且恢复新活动数据库的日志传送，以便接收新消息。

切换活动数据库和备用数据库之后

了解所涉及的过程以及在从活动数据库切换到备用数据库后热备份环境中组件的状态。

在已经切换活动数据库和备用数据库的角色之后，复制系统将发生更改，如下图所示：

图 8： 热备份应用程序示例 - 切换之后



- 以前的备用数据库现在是新的活动数据库。客户端应用程序将切换到新的活动数据库。
- 在本示例中，以前的活动数据库现在成为新的备用数据库。以前的活动数据库的消息将依次排队，以便应用到新的活动数据库。

注意： 切换之后，以前的活动数据库的 Replication Agent 关闭，新的活动数据库的 Replication Agent 启动。

进行切换

从活动数据库切换到备用数据库包括几个任务。

1. 如果客户端应用程序仍在使用活动数据库，则断开这些应用程序与活动数据库的连接

2. 在 Replication Server 中，切换活动数据库和备用数据库
3. 通过新的活动数据库重新启动客户端应用程序
4. 启动新的活动数据库的 RepAgent
5. 确定是删除旧的活动数据库还是将它用作新的备用数据库

断开客户端应用程序与活动数据库的连接

在切换到备用数据库之前，必须让客户端停止执行活动数据库中的事务。

当然，如果数据库出现故障，客户端就不能执行事务。不过，您需要执行一些步骤来防止客户端在数据库恢复联机之后对其进行更新。

另请参见

- 设置客户端以使用活动数据服务器 （第 84 页）

切换活动数据库和备用数据库

了解切换逻辑连接的活动数据库和备用数据库的过程。

前提条件

在切换之前，您必须将客户端设置为使用活动数据服务器。

过程

1. 在活动数据库的 Adaptive Server 上，确保 RepAgent 已关闭。否则，请使用 **sp_stop_rep_agent** 关闭 RepAgent。
2. 在 Replication Server 上执行 switch active 命令。

   输入：

   ```
   switch active for logical_ds.logical_db
   to data_server.database
   ```

   data_server.database 是新的活动数据库。
3. 使用 **admin logical_status** 监控切换的进度。

   输入：

   ```
   admin logical_status, logical_ds, logical_db
   ```

   有关切换状态，请参见 `Operation in Progress` 和 `State of Operation in Progress` 输出列。
4. 活动数据库切换完成后，必须重新启动新活动数据库的 RepAgent。

   输入：

   ```
   sp_start_rep_agent dbname
   ```

下一

注意： 如果 Replication Server 在切换过程中停止，那么，在您重新启动 Replication Server 后，切换会继续进行。

另请参见

- 设置客户端以使用活动数据服务器 （第 84 页）
- 内部切换步骤 （第 77 页）

在切换活动数据库时使用阻塞命令

可以使用 **wait for switch** Replication Server 阻塞命令指示 Replication Server 等待，直到备用数据库做好操作准备。

您可以在切换活动数据库的脚本中使用此命令。语法为：

```
wait for switch for logical_ds.logical_db
```

监控切换

可以使用 **admin logical_status** 来检查是否存在阻止切换继续的复制系统问题。

备用数据库的事务日志被写满或备份 DSI 被挂起，这些都属此类问题。如果不能解决这些问题，则可使用 **abort switch** 命令中止切换。

`Operation in Progress` 和 `State of Operation in Progress` 输出列指明切换的状态。

例如，假定 **admin logical_status** 总是在其 `State of Operation in Progress` 输出列中返回以下消息之一：

```
Standby has some transactions that have not been applied
```

或者

```
Inbound Queue has not been completely read by Distributor
```

这些消息表明可能出现了无法解决的问题，在此情况下，您可以选择中止切换。使用 **admin who** 命令即可获得有关切换操作状态的详细信息。

另请参见

- 监控热备份应用程序的命令 （第 83 页）

中止切换

除非 Replication Server 在切换活动数据库和备用数据库的过程中已经继续得太远，否则可以使用 **abort switch** 命令中止该过程。

语法为：

```
abort switch for logical_ds.logical_db
```

如果 **abort switch** 命令成功取消 **switch active** 命令，您可能需要重新启动活动数据库的 RepAgent。

当 **switch active** 命令执行到一定的阶段后，您就不能取消该命令。在此情况下，您必须等待 **switch active** 命令完成，然后再次使用该命令返回到原来的活动数据库。

重新启动客户端应用程序

如果 **admin logical_status** 命令表明没有正在进行的操作，或者 **wait for switch** 命令返回 **isql** 提示符，则可在新的活动数据库中重新启动客户端应用程序。

客户端应用程序必须等待 Replication Server 切换到新的活动数据库之后，才能开始在新的活动数据库中执行事务。您应该提供一种有序的方法，将客户端从旧的活动数据库移到新的活动数据库中。

另请参见

- 设置客户端以使用活动数据服务器 （第 84 页）

解决残余事务

如果旧的活动数据库出现故障，您应该确定是否有事务尚未传送到新的活动数据库中。如果有执行这样的事务的外部记录，这些事务就被称为 *“残余事务”*。

从活动数据库切换到备用数据库时，在切换完成之前，入站队列中所有提交的事务会应用于新的活动数据库。不过，出现故障之前在活动数据库中提交的某些事务可能未被 Replication Server 接收，因此，这些事务不会应用到备用数据库中。

切换活动数据库和备用数据库时，您可以重新执行新的活动数据库中的残余事务。如果存在依赖性，则可能需要先重新执行残余事务，然后才能允许执行新事务。一定要使用原始客户端的登录名（而不是维护用户的登录名）执行残余事务。

如果您将旧的活动数据库作为新的备用数据库联机，必须先颠倒残余事务的顺序，这样才不会在备用数据库中复制这些残余事务。

管理旧的活动数据库

切换到新的活动数据库后，您必须决定如何处理旧的活动数据库。

您可以：

- 将数据库作为新的备用数据库联机并恢复连接，这样 Replication Server 就能应用新事务，或者
- 使用 **drop connection** 删除数据库连接，并随后再次添加它作为新的备用数据库。如果删除数据库，队列中与该数据库有关的所有消息都将被删除。请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**drop connection**”。

将旧的活动数据库作为新的备用数据库联机

如果旧的活动数据库未被损坏，则可以将其作为新的备用数据库联机。

输入：

```
resume connection to data_server.database
```

其中 *data_server.database* 是旧的活动数据库的物理数据库名称。

为避免出现重复事务，您可能需要解决数据库中的残余事务。根据您的应用程序，您可能需要在将旧的活动数据库作为新的备用数据库联机之前解决残余事务。

因为必须在新的活动数据库中重新执行残余事务，所以您必须准备好新的备用数据库，以便再次接收通过复制系统传送的残余事务。

要解决此冲突，您可以：

- 撤消新的备用数据库中的重复事务或颠倒它们的顺序，或者
- 忽略重复事务，以后再处理。

监控热备份应用程序

可以使用 Replication Server 日志文件或一些命令监控两个 Adaptive Server 数据库或 Oracle 数据库之间的热备份应用程序。

Replication Server 日志文件

您可以在 Replication Server 日志文件中查看与热备份操作有关的消息，例如，在添加备用数据库时看到的消息。

已创建备份连接

下面是 Replication Server 在创建备用数据库的物理连接时写入的消息示例：

```
I. 95/11/01 17:47:50. Create starting : SYDNEY_DS.pubs2
I. 95/11/01 17:47:58. Placing marker in TOKYO_DS.pubs2 log
I. 95/11/01 17:47:59. Create completed : SYDNEY_DS.pubs2
```

在这些示例中，SYDNEY_DS 是备份数据服务器，TOKYO_DS 是活动数据服务器。

在为备用数据库创建物理连接时，Replication Server 在活动数据库事务日志中写入“启用复制”标记。备份 DSI 在收到此标记前忽略所有事务。不过，如果选择“转储标记”选项，备份 DSI 在遇到日志中的下一个转储标记之前会继续忽略消息。

当相应的标记从活动数据库的 Replication Agent 到达备用数据库时，备份 DSI 将在 Replication Server 日志文件中写入一条消息，然后开始执行备用数据库中的后续事务。

在上面的消息示例中，Replication Server 已经为备用数据库 SYDNEY_DS.pubs2 创建了连接，并挂起了其 DSI 线程。此时，数据库管理员将转储活动数据库 TOKYO_DS.pubs2 的内容，并将其装载到备用数据库中。

初始化后恢复备份连接

数据库管理员将转储装载到备用数据库，并且恢复与备用数据库的连接后，备份 DSI 开始处理来自活动数据库的消息。Replication Server 在其日志中写入类似下面的消息：

```
I. 95/11/01 18:50:34. The DSI thread for database 'SYDNEY_DS.pubs2'
is started.
I. 95/11/01 18:50:41. Setting LTM truncation to 'ignore' for
```

```
SYDNEY_DS.pubs2 log
I. 95/11/01 18:50:43. DSI for SYDNEY_DS.pubs2 received and processed
Enable
    Replication Marker. Waiting for Dump Marker
I. 95/11/01 18:50:43. DSI for SYDNEY_DS.pubs2 received and processed
Dump
    Marker. DSI is now applying commands to the Standby
```

当您在日志文件中看到最终的消息时，说明热备份数据库的创建进程已完成。

监控热备份应用程序的命令

admin 命令用于监控热备份应用程序的状态。

有关这些命令的详细信息，请参见《Replication Server 参考手册》中的“Replication Server 命令”。

admin logical_status

admin logical_status 命令显示：

- 添加备用数据库的进展状态，或者，切换活动数据库与备用数据库的进展状态。
- 活动数据库或备用数据库连接是否被挂起。
- 备份 DSI 是否忽略消息。在等待来自活动数据库的标记到达事务流的过程中，备份 DSI 将忽略消息。

admin who, dsi

admin who, dsi 命令提供了另一种检查备份 DSI 状态的方法。`IgnoringStatus` 输出列包含以下内容之一：

- “Applying” – 如果 DSI 将消息应用于备用数据库，或
- “Ignoring” -- 如果 DSI 正在等待标记。

admin who, sqm

admin who, sqm 命令提供有关稳定队列的状态消息。在热备份应用程序中，由分配器线程（如果您尚未禁用它）和备份 DSI 线程读取入站队列。要等这两个线程都读取并且传送了来自入站队列的消息后，Replication Server 才能删除这些消息。

如果 Replication Server 不删除来自入站队列的消息，您可以使用 **admin who, sqm** 命令来调查问题。您将从该命令获悉有多少线程正在读取队列，以及队列中的最小删除点。

admin sqm_readers

admin sqm_readers 命令监控正在读取入站队列的单个线程的读取和删除点。如果不删除入站队列，**admin sqm_readers** 将有助于您查找没有读取该队列的线程。

admin sqm_readers 命令使用两个参数：逻辑连接的队列号和队列类型。

您可以在 **admin who, sqm** 命令输出的 `Info` 列中找到队列号和队列类型：队列号是冒号左边的 3 位数数字，队列类型是冒号右边的数字。

队列类型 1 是入站队列。队列类型 0 是出站队列。逻辑连接的入站队列可以由多个线程读取。例如，要查找读取入站队列号 102 的线程，请执行下面的 **admin sqm_readers** 命令：

```
admin sqm_readers, 102, 1
```

设置客户端以使用活动数据服务器

在 Replication Server 中使用 **switch active** 命令切换活动数据库和备用数据库时，您必须借助一种方法来切换客户端应用程序，因为 Replication Server 不会自动将客户端应用程序切换到新的活动数据服务器和数据库。

可以通过三种示例方法设置客户端应用程序以连接到当前的活动数据服务器。您可以创建：

- 两个 interfaces 文件
- 一个 interfaces 文件条目，带有客户端应用程序的数据服务器符号名
- 一种机制，自动将客户端应用程序的数据服务器连接映射到当前的活动数据服务器

必须在设置热备份数据库之前实现您的方法。

无论使用哪种方法来切换应用程序，请勿修改 Replication Server 使用的 interfaces 文件条目。

两个 interfaces 文件

使用此方法，可以设置两个 interfaces 文件，一个用于客户端应用程序，一个用于 Replication Server。

您在切换客户端时，可以修改它们的 interfaces 文件条目，以便在新的活动数据库中使用数据服务器的主机名和端口号。

客户端应用程序的数据服务器符号名

通过此方法创建一个 interfaces 文件条目，它带有客户端应用程序的数据服务器符号名。

interfaces 文件可能包含数据服务器名、主机名和端口号的条目。

表 10. interfaces 文件中的数据服务器符号名

	数据服务器名	主机名	端口号
客户端应用程序	CLIENT_DS	*machine_1*	2800
活动数据库	TOKYO_DS_X	*machine_1*	2800
备用数据库	TOKYO_DS_Y	*machine_2*	2802

您可以为名为 CLIENT_DS 的数据服务器创建接口条目。客户端应用程序将始终连接到 CLIENT_DS.CLIENT_DS 条目使用的主机名和端口号与连接到活动数据库的数据服务器所用的主机名和端口号相同。

Replication Server 连接到的主机名和端口号与客户端应用程序连接到的主机名和端口号相同，但它们使用不同的数据服务器名。在本示例中，Replication Server 将在 TOKYO_DS_X 与 TOKYO_DS_Y 数据服务器之间切换。

切换活动数据库后，您应将 CLIENT_DS 接口条目更改为连接到新的活动数据库的数据服务器所用的主机名和端口号，在本示例中，主机名是 *machine_2*，端口号是 2802。

将客户端数据服务器映射到当前的活动数据服务器

通过此方法，您可以创建一种机制（如中间 Open Server 应用程序），该机制可以自动将客户端应用程序数据服务器连接映射到当前活动数据服务器。

有关如何创建此类 Open Server 应用程序的详细信息，请参见 Open Server 文档，例如《Open Server Server-Library/C 参考手册》。

更改热备份数据库连接

了解用于重新配置或修改逻辑数据库连接和物理数据库连接的选项。通常情况下，如果通过正常过程设置热备份应用程序，缺省设置会正常起作用。

更改逻辑连接

可以使用 **alter logical connection** 命令修改热备份逻辑数据库连接的参数。

可以使用 **alter logical connection** 命令修改以下参数：

- 影响逻辑连接的参数
- 启用或禁用分配器线程的参数
- 启用或禁用将 **truncate table** 复制到备用数据库的参数

更改影响逻辑连接的参数

可以使用 **alter logical connection** 命令更新影响逻辑连接的参数。

登录到源 Replication Server，然后在 **isql** 提示符下输入：

```
alter logical connection
 to logical_ds.logical_db
set logical_database_param to 'value'
```

其中 *logical_ds* 是逻辑连接的数据服务器名，*logical_db* 是逻辑连接的数据库名，*logical_database_param* 是逻辑数据库参数，*value* 是该参数的字符串设置。

新设置会立即生效。

影响逻辑连接的配置参数

您可以使用多个参数来配置逻辑连接。

警告！ 只有在稳定队列空间严重不足时，才应该重新设置逻辑连接参数 **materialization_save_interval** 和 **save_interval**。如果将这些参数从 **strict** 重新设置为给定的分钟数，备用数据库中的消息可能会丢失。

表 11. 影响逻辑连接的配置参数

logical_database_param	*value*
deferred_name_resolution	在 Replication Server 中启用延迟名称分析以便在 Adaptive Server 中支持延迟名称分析。 在 Replication Server 中启用延迟名称分析支持之前，您必须确保复制 Adaptive Server 中支持延迟名称分析。 缺省值：off **注意：** 此参数仅适用于 Adaptive Server。
materialization_save_interval	实现队列保存间隔。此参数仅用于热备份应用程序中的备用数据库。 缺省值：“strict”（对于备用数据库）
replicate_minimal_columns	指定 Replication Server 是发送所有事务的所有复制定义列，还是仅发送在备用数据库中执行更新或删除操作所需要的复制定义列。值为“on”和“off”。 仅在复制定义不包含“send standby”参数或根本没有复制定义时，Replication Server 才在备用情况下使用该值。 否则，Replication Server 将使用复制定义中的“replicate minimal columns”或“replicate all columns”参数的值。 缺省值：on 在将 **dsi_compile_enable** 设置为“on”时，Replication Server 会忽略您为 **replicate_minimal_columns** 设置的内容。
save_interval	指定保存间隔，这是将消息成功传送到目标数据服务器后 Replication Server 对其进行保存的分钟数。 缺省值：0 分钟

logical_database_param	*value*
send_standby_repdef_cols	指定 Replication Server 应将哪些列发送到备用数据库以建立逻辑连接。替换复制定义中的“send standby”选项，此选项用于通知 Replication Server 要将哪些表列发送到备用数据库。值为： • **on** – 只发送匹配复制定义中出现的表列。忽略复制定义中的“send standby”选项。 • **off** – 将所有表列发送到备用数据库。忽略复制定义中的“send standby”选项。 • **check_repdef** – 根据“send standby”选项，将所有表列发送到备用数据库。 缺省值：**check_repdef**

另请参见

- 恢复的保存间隔 （第 286 页）

禁用分配器线程

可以使用 **alter logical connection** 命令禁用分配器线程。

如果不将数据从活动数据库复制到备用数据库以外的其它数据库，Replication Server 则不需要对逻辑连接使用分配器线程。您可以禁用分配器线程以节省 Replication Server 资源。

若要禁用分配器线程，必须首先删除对逻辑数据库中的数据的所有预订。然后，在 Replication Server 上执行 **alter logical connection**：

```
alter logical connection
 to logical_ds.logical_db
set distribution off
```

如果以后决定要复制活动数据库中的数据，则可使用此命令重新启用分配器线程。

警告！ 如果禁用分配器线程，然后从复制系统中删除备用数据库，则不会留下任何 Replication Server 线程来从活动数据库读取入站队列。在您添加另一个备用数据库并将逻辑连接的分配设为“on”，或者从复制系统删除活动数据库之前，入站队列将一直填充。

将 truncate table 复制到备用数据库

可以使用 **alter logical connection** 命令启用或禁用 **truncate table** 命令复制。

Replication Server 会将 **truncate table** 的执行复制到热备份数据库。活动数据库和备用数据库必须是 Adaptive Server 11.5 版或更高版本才能支持此功能。

要启用或禁用 **truncate table** 的复制，请登录到源 Replication Server 并输入：

```
alter logical connection
 to logical_ds.logical_db
set send_truncate_table to {on | off}
```

如果热备份应用程序是在安装或升级到 Replication Server 11.5 版或更高版本*之前*创建的，除非使用 **alter logical connection** 启用此功能，否则 Replication Server 不会将 **truncate table** 复制到备用数据库。为了保持与现有热备份应用程序的兼容性，缺省设置为“off”。

如果热备份应用程序是在安装或升级到 Replication Server 11.5 版或更高版本*之后*创建的，除非使用 **alter logical connection** 禁用此功能，否则 Replication Server 会自动将 **truncate table** 复制到备用数据库。缺省设置为“on”。

更改物理连接

您可以在源 Replication Server 上使用 **alter connection** 命令修改影响热备份应用程序的物理连接的参数。

语法为：

```
alter connection to data_server.database
set database_param to 'value'
```

其中 *data_server* 是目标数据服务器，*database* 是数据服务器管理的数据库，*database_param* 是影响连接的参数，*value* 是 *database_param* 的设置。

您必须先挂起连接，然后才能对它进行更改；在执行了 **alter connection** 命令之后，请恢复该连接，以使新参数设置生效。请参见《Replication Server 管理指南第一卷》的“管理数据库连接”中的“更改数据库连接”。

配置备用数据库中的触发器

可以使用 **alter connection** 命令配置连接以引发或不引发触发器。

缺省情况下，备用 DSI 线程在登录到备用数据库时将执行 **set triggers off** Adaptive Server 命令。这样可防止 Adaptive Server 引发复制的事务的触发器，从而防止在备用数据库中发生重复更新。

通过使用 **alter connection** 命令配置连接以激发或不激发触发器，可以更改缺省行为。为此，请将 **dsi_keep_triggers** 配置参数设置为“on”或“off”。对于备用数据库以外的所有连接，**dsi_keep_triggers** 缺省设置均为“on”。

配置备用数据库中的复制

设置 **dsi_replication** 配置参数以指定由 DSI 应用的事务在事务日志中是否标记为要复制。

对于活动的复制数据库，该参数必须设为“on”。缺省情况下，对于备用数据库，该参数设为“off”；对于所有其它数据库，该参数设为“on”。

如果将 **dsi_replication** 设置为“off”，则 DSI 将在数据库中执行 **set replication off**，从而使 Adaptive Server 无法将复制信息添加到 DSI 所执行的事务的日志记录中。由于这些事务由维护用户执行，因此不会进一步复制（除非存在备用数据库），在合适的情况下将该参数设为“off”可减少写入事务日志的信息量。

使用 **admin who, dsi** 命令可查看此参数针对某个连接的具体设置。

更改备用数据库中的配置参数

如果为活动数据库设置了以下配置参数，那么在创建备用数据库时，一些参数将从活动数据库复制到备用数据库。您可以更改上述任何配置参数的设置。

表 12. 复制到备用数据库的配置参数

batch	**batch_begin**	**command_retry**
db_packet_size	**dsi_cmd_separator**	**dsi_charset_convert**
dsi_cmd_batch_size	**dsi_keep_triggers**	**dsi_fadeout_time**
dsi_isolation_level	**dsi_max_text_to_log**	**dsi_large_xact_size**
dsi_max_cmds_to_log	**dsi_replication**	**dsi_num_large_xact_threads**
dsi_num_threads	**dsi_xact_group_size**	**dsi_serialization_method**
dsi_sqt_max_cache_size	**dsi_xact_in_group**	**dump_load**
parallel_dsi	**use_batch_markers**	

请参见《Replication Server 管理指南第一卷》中的“管理数据库连接”。

删除逻辑数据库连接

如果要卸除热备份应用程序，您可能需要从复制系统删除逻辑数据库。

为此，请删除备用数据库，然后执行 **drop logical connection** 命令。在执行该命令前，必须删除备用数据库。有关删除物理数据库连接的信息，请参见《Replication Server 管理指南第一卷》的“管理数据库连接”中的“删除数据库连接”。

drop logical connection 的语法是：

```
drop logical connection to data_server.database
```

data_server 和 *database* 表示逻辑数据服务器和逻辑数据库。

例如，要删除与 LDS 逻辑数据服务器中的 `pubs2` 逻辑数据库的连接，请输入：

```
drop logical connection to LDS.pubs2
```

从 ID Server 中删除逻辑数据库

当热备份应用程序存在于复制系统中时，逻辑数据库（连同物理数据库、数据服务器和复制服务器一起）将在 ID Server 的 RSSD 中的 `rs_idnames` 系统表中列出。有时，可能需要从该系统表中删除某个逻辑数据库的条目。

例如，如果 **drop logical connection** 命令失败，您可能必须强制 ID Server 从 `rs_idnames` 系统表中删除对应于逻辑数据库的行。逻辑数据库连接在 `ltype` 列中显示“L”。

sysadmin dropldb 命令登录到 ID Server，并删除指定逻辑数据库的条目。语法为：

```
sysadmin dropldb, data_server, database
```

data_server 和 *database* 指逻辑数据服务器名和逻辑数据库名。

您必须有 **sa** 权限，才能执行任何 **sysadmin** 命令。

使用复制的热备份应用程序

了解涉及到复制的热备份应用程序，其中，逻辑数据库作为复制系统中的主数据库或复制数据库。

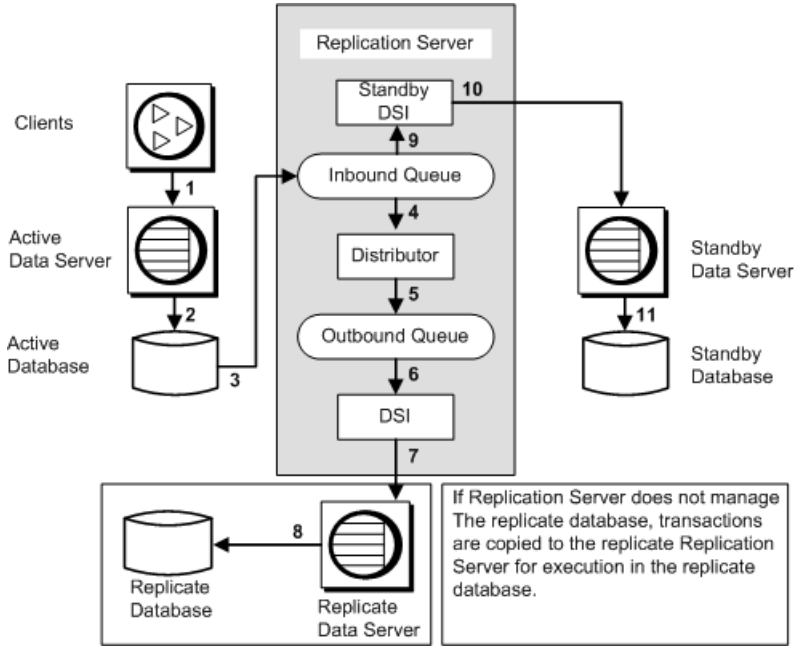
主数据库的热备份应用程序

了解主数据库的热备份应用程序。

此图说明主数据库的热备份应用程序示例。在本示例中，一个 Replication Server 管理三个数据库：

- 逻辑主数据库的活动数据库
- 逻辑主数据库的备用数据库，以及
- 复制数据库，它具有对逻辑主数据库中数据的预订。

图 9： 主数据库的热备份应用程序



在本示例中，一个 Replication Server 既管理主数据库又管理复制数据库。其它情况下，可能由不同的 Replication Server 管理主数据库和复制数据库。

图中的数字表示从客户端应用程序通过主数据库的热备份应用程序中的复制系统的事务流。

从客户端应用程序到入站队列
在该图中，数字 1 到 3 表示从客户端到 Replication Server 中的入站队列的事务：

- 客户端执行活动的主数据服务器中的事务。
- 活动的主数据服务器更新活动的主数据库。
- 活动主数据库的 Replication Agent 从数据库日志读取复制数据的事务。它将事务转发到 Replication Server，后者再将这些事务写入入站队列。
  复制的数据的所有事务（包括由维护用户执行的事务）都会发送到 Replication Server，以便应用到备用数据库中。

从入站队列到复制数据库
在该图中，数字 4 到 8 将跟踪从入站队列到复制数据库的事务：

- 分配器线程从入站队列读取事务。
- 分配器线程针对预订处理事务，并将复制的事务写入出站队列。
  由于您在使用 **sp_config_rep_agent** 配置 RepAgent 时设置了 **send_warm_standby_xacts** 参数，所以，维护用户执行的事务总是会复制到备用数据库，但不会复制到复制数据库，除非您还为 RepAgent 设置了 **send_maint_xacts_to_replicate**。

  注意： 对于 Oracle，始终将维护用户执行的事务复制到复制数据库，因为 **filter_maint_userid** 配置参数对 Replication Agent for Oracle 无效，与该参数设置为“true”还是“false”无关。
- DSI 线程从出站队列读取事务。
- DSI 线程执行复制数据服务器中的事务。
- 复制数据服务器更新复制数据库。
  如果事务要复制到由不同的 Replication Server 管理的数据库，它们将写入由 RSI 线程（而非 DSI 线程）管理的 RSI 出站队列中。RSI 线程将事务传送到其它 Replication Server。

从入站队列到备用数据库
在该图中，数字 9 到 11 将跟踪从入站队列到逻辑主数据库的备用数据库的事务：

- 备份 DSI 线程从入站队列读取事务。
- 备份 DSI 线程执行备份数据服务器中的事务。
- 备份数据服务器更新备用数据库。

入站队列由备份 DSI 和分配器读取。这两个线程会并发工作。在这两个线程都读取了来自入站队列的消息并将它们传送到其目标之前，不能从入站队列中截断这些消

息。这些消息将一直保留在队列中，直到 DSI 将它们应用于备用数据库；而且，如果有预订或复制的存储过程执行，分配器会将它们写入出站队列。

取决于具体的复制系统，事务可能先复制到备用数据库中，然后再到复制数据库。不过，Replication Server 保证备份主数据库和复制数据库与活动主数据库保持同步。

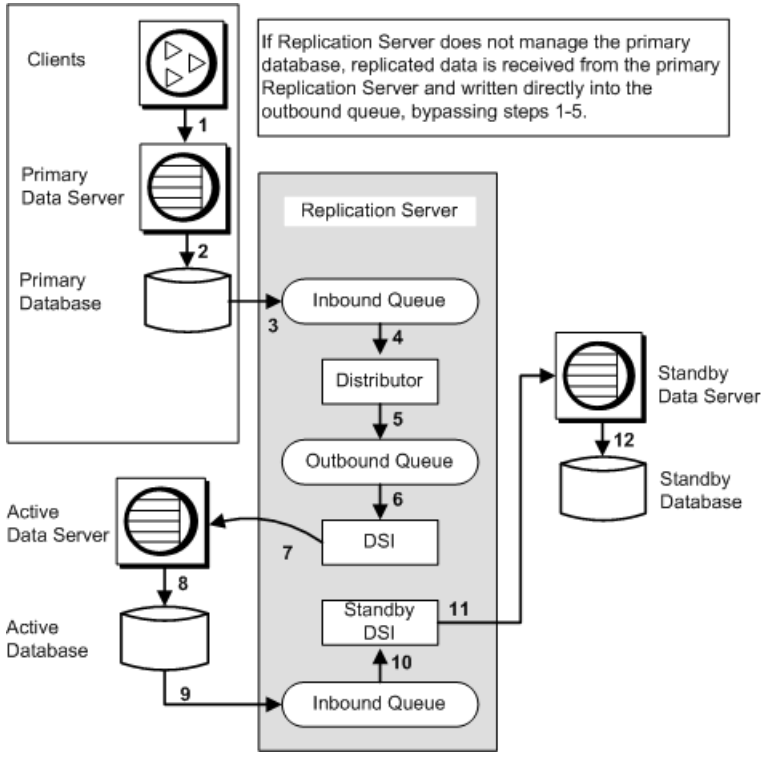
复制数据库的热备份应用程序

了解复制数据库的热备份应用程序。

此图说明复制数据库的热备份应用程序示例。在本示例中，一个 Replication Server 管理三个数据库：

- 主数据库，
- 逻辑复制数据库的活动数据库，以及
- 逻辑复制数据库的备用数据库

图 10： 复制数据库的热备份应用程序



逻辑复制数据库具有对主数据库中数据的预订。因此，来自主数据库的更新将复制到活动数据库和备用数据库。

在本示例中，一个 Replication Server 既管理主数据库又管理复制数据库。其它情况下，可能由不同的 Replication Server 管理主数据库和复制数据库。

此图中的数字表示从客户端应用程序通过复制数据库的热备份应用程序中的复制系统的事务流。

从客户端应用程序到主数据库和活动数据库
在该图中，数字 1 到 8 跟踪从客户端到主数据库并通过正常复制转到活动复制数据库的事务：

- 客户端执行主数据服务器中的事务。
- 主数据服务器更新主数据库。
- 主数据库的 Replication Agent 从事务日志读取复制的数据的事务，并将事务转发到 Replication Server，后者再将这些事务写入入站队列。
- 分配器线程从入站队列读取事务。
- 分配器针对预订处理事务，并将复制的事务写入出站队列。
  如果管理复制数据库的热备份应用程序的 Replication Server 并不管理主数据库，则从主 Replication Server 接收复制的数据，并将它们直接写入出站队列，绕过第 1 步到第 5 步。
- DSI 线程从出站队列读取事务。
- DSI 线程执行复制数据服务器中的事务，该服务器是热备份应用程序的活动数据服务器。
- 活动数据服务器更新活动数据库。
  如果事务源于由不同的 Replication Server 管理的主数据库，主 Replication Server 中的分配器线程会将它们写入 RSI 出站队列。随后，这些事务被复制到复制 Replication Server 中的 DSI 出站队列，以便应用于逻辑复制数据库的活动数据库。

从活动数据库到备用数据库
在该图中，数字 9 到 12 跟踪从逻辑复制数据库的活动数据库到其备用数据库的事务：

- 活动数据库的 Replication Agent 从活动数据库日志读取事务，并将事务转发到 Replication Server，后者再将这些事务写入入站队列。
  复制的数据的所有事务（包括由维护用户执行的事务）都会发送到 Replication Server，以便应用到备用数据库中。
- 备份 DSI 线程从入站队列读取事务。
- 备份 DSI 线程执行备份数据服务器中的事务。
- 备份数据服务器更新备用数据库。

配置逻辑连接保存间隔

可以使用 DSI 队列保存间隔或实现队列保存间隔，为逻辑复制数据库重新配置保存间隔。

连接的保存间隔指定消息被删除之前保留在稳定队列中的时间。如果您通过正常过程设置热备份应用程序，缺省设置会正常起作用。

可以使用 **configure logical connection** 配置逻辑连接的 DSI 队列保存间隔和实现队列保存间隔。

请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**configure logical connection**”。

警告！ 只有 在缺少稳定队列空间这样的严重情况下，才能重新设置 DSI 队列保存间隔和实现队列保存间隔。如果将这些保存间隔从 **strict** 重新设置为给定的分钟数，备用数据库中的消息可能会丢失。Replication Server 检测不到此类丢失，您必须自行验证备用数据库的完整性。

DSI 队列保存间隔

缺省情况下，当您创建备用数据库时，逻辑连接的 DSI 队列保存间隔被设置为 **'strict'**。

这样，Replication Server 将一直保留 DSI 队列消息，直到它们被传送到备用数据库。如果您必须更改逻辑连接的 DSI 队列保存间隔，应使用 **configure logical connection** 命令。

例如，若要强制复制 Replication Server 在持续一个小时（六十分钟）的时间内保存发往其逻辑复制数据服务器 LDS 的消息，请输入以下命令：

```
configure logical connection to LDS.logical_pubs2
set save_interval to '60'
```

要将此保存间隔重新设置为 **'strict'**，则输入：

```
configure logical connection to LDS.logical_pubs2
set save_interval to 'strict'
```

实现队列保存间隔

缺省情况下当您创建预订时，逻辑连接的实现队列保存间隔被设置为 **'strict'**。

这样，Replication Server 将一直保留实现队列消息，直到它们被传送到备用数据库。如果您必须更改逻辑连接的实现队列保存间隔，应使用 **configure logical connection** 命令。

例如，若要强制复制 Replication Server 在持续一个小时（六十分钟）的时间内保存其逻辑复制数据服务器 LDS 的实现队列中的消息，请输入以下命令：

```
configure logical connection to LDS.logical_pubs2
set materialization_save_interval to '60'
```

要将此保存间隔重新设置为 **'strict'**，则输入：

```
configure logical connection to LDS.logical_pubs2
set materialization_save_interval to 'strict'
```

热备份数据库的复制定义和预订

您可以为热备份应用程序创建复制定义和预订。

在仅包含 Adaptive Server 数据库的复制系统中，如果复制定义的唯一目的是指定主键、带引号的标识符信息或自定义函数字符串，则无需在热备份环境或多节点可用性 (MSA) 环境中使用表的复制定义。请参见《Replication Server 管理指南第一卷》的“使用多节点可用性管理复制对象”中的“减少复制定义和预订的使用”。

不过，您可以为逻辑数据库中的每个表创建复制定义。也可以在复制到备用数据库时使用函数复制定义。复制定义可以更改 Replication Server 将数据复制到备用数据库的方式，这样，您可以借助复制定义来优化热备份应用程序，或启用应用程序所需的非缺省行为。

在进出逻辑数据库的正常复制过程中，可以在热备份应用程序中使用复制定义。

另请参见

- 使用复制的热备份应用程序 （第 90 页）

热备份的 alter table 支持

Adaptive Server Enterprise 版本 12.0 和更高版本允许用户更改现有表，即添加不可为 Null 的列、删除列和修改列数据类型。

对于 Oracle 热备份应用程序，您需要使用复制定义启用用户定义的数据类型复制。Replication Agent for Oracle 在初始化时自动创建复制定义。在这种情况下，您需要手动创建新的复制定义或更改现有的复制定义，以便在复制定义中显式指定将哪些用户定义的数据类型复制到备用数据库。

在表没有预订的情况下，Replication Server 支持 **alter table** 命令所导致的表更改。

注意： 若要在存在该表的预订时支持来自于 **alter table** 的表更改，则需要改变表的复制定义。有关说明，请参见《Replication Server 管理指南第一卷》的“管理复制表”中的“修改复制定义”。

在先前的版本中，如果为表定义了复制定义，Replication Server 总是会使用在热备份复制定义中定义的列数据类型。从 Replication Server 12.0 版开始，视具体情况的不同，Replication Server 可能使用也可能不使用表的复制定义。

无复制定义

在对没有复制定义的表使用 **alter table** 命令时，Replication Server 向热备份数据库发送的信息与它从主服务器接收的信息相同。

支持 **alter table** 的所有选项。在主数据库执行 **alter table** 时，该命令被复制到热备份数据库，并且继续执行向备用数据库复制数据的操作，而 Replication Server 中无需任何操作。

请参见《Adaptive Server Enterprise 参考手册：命令》的“命令”中的“**alter table**”以了解相应的语法和信息。

alter table 添加带缺省值的列

了解在活动数据库中发出 **alter table** 命令以添加带缺省值的列时如何避免 DSI 错误。

当您在活动数据库中发出 alter table 命令以添加使用缺省值的列时，Adaptive Server 将以自动生成的名称创建约束。如果将该命令复制到备用数据库，备用数据库也会创建相同的约束，但自动生成另一个不同的约束名。在活动数据库中删除约束时，备用数据库将不识别约束名，并生成数据服务器接口 (DSI) 错误。

为避免出现这种情况，请先删除活动数据库中的约束。数据服务器接口 (DSI) 自动关闭。之后，删除在备用数据库中创建的约束，然后发出 **resume dsi skip transaction** 命令。

另一解决方法是执行：

```
alter table table name
replace column name
default null
```

这将自动删除在活动节点和备份节点创建的约束。

不带 send standby 子句的热备份

如果没有与任何复制定义关联的 **send standby** 子句，Replication Server 将发送它从主表收到的所有数据，而不参照复制定义。

Replication Server 使用原始列名和数据类型发送从 Replication Agent 收到的数据。复制定义仅用于查找主键。主键是表的所有复制定义中的主键集合。

如果模式更改不包括删除表的所有复制定义中的所有主键列，则情况与没有复制定义时相同。支持 **alter table** 的所有选项，并且不需要在 Replication Server 中执行任何操作。

在更改主表之前，您随时可以通过更改复制定义来删除复制定义中的所有主键，并将新的主键列添加到复制定义中。

只有在从复制系统清除所有旧数据行后，才能删除旧主键。否则，“数据服务器接口”(DSI) 将关闭。如果出现此情况，请参见恢复说明。

另请参见

- 无复制定义 （第 95 页）

带有 send standby all columns 子句的热备份

如果 **send standby all columns** 与某个复制定义关联，Replication Server 将发送它从 Replication Agent 收到的所有数据，并使用原始列名和数据类型。复制定义仅用于查找主键。

如果模式更改不涉及使用 send standby all columns 子句在复制定义中删除所有主键列，则情况与没有复制定义时相同。支持 **alter table** 的所有选项，并且不需要在 Replication Server 中执行任何操作。

在更改主表之前，您随时可以通过更改复制定义来删除带有 **send standby all columns** 子句的复制定义中的所有主键，并将新的主键列添加到复制定义中。

另请参见

- 无复制定义 （第 95 页）

带有 send standby replication definition columns 子句的热备份

如果在复制定义中有 **send standby replication definition columns** 子句，则备用数据库将继续使用在表的相应复制定义中所定义的复制表名称和列名称以及数据类型。

如果您要在备份中使用复制定义数据类型，总是要创建带有 **send standby replication definition columns** 子句的复制定义。

使用复制定义优化性能

可以使用复制定义提高热备份环境中的复制系统的性能。

当您指定在向备用数据库复制数据的过程中使用复制定义时：

- 可以指定 Replication Server 是否要对复制到备用数据库中的过程使用复制定义的 **“复制最少列”** 设置。该设置指明更新操作是替换所有列的值，还是只替换更改了值的那些列的值。
- 可以指定 Replication Server 是将表的所有列或存储过程的所有参数复制到备用数据库，还是只将表或函数复制定义中列出的那些列或参数复制到备用数据库。

为复制到备用数据库的操作创建复制定义

如果只是为复制到备用数据库的操作创建复制定义，请在 **create replication definition** 命令中使用 **send standby** 子句。

在复制到备用数据库时，将使用复制定义的主键和 **“复制最少列”** 设置。

请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**create replication definition**”。

指定主键

表复制定义是否存在决定了如何在数据库的 **where** 子句中打包主键列。请参见《Replication Server 管理指南第一卷》的“使用多节点可用性管理复制对象”中的“减

少复制定义和预订的使用”下面的“热备份和多节点可用性环境中的主键列和 **where** 子句打包”。

更新最少列数
当您为复制到备用数据库的操作创建复制定义时，可以利用另一种复制系统性能优化方法，即最少列数设置。

如果使用 **replicate minimal columns** 子句，复制的 **update** 和 **delete** 事务只包含必需的列。未更改的列值可以从 **update** 命令中省略。如果省略不必要的列，则可减小通过复制系统传送的消息大小，并能减少 Adaptive Server 的工作量。

如果在向备用数据库复制数据的过程中不使用复制定义，仍可获得此性能优势。

如果表没有复制定义，或者如果有复制定义但在复制到备用数据库的过程中不使用复制定义，则会自动复制最少的列数。

指定要复制到备用数据库的列
在为复制到备用数据库的操作创建复制定义时，您可以指定要复制的列集。

- 指定 **send standby** 或 **send standby all columns** 以便将表中所有列复制到备用数据库中。
- 指定 **send standby replication definition columns** 可以只将复制定义的列复制到备用数据库中。

有关在该命令中使用 **send standby** 子句的详细信息，请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**create replication definition**”。

指定要复制到备用数据库的参数
在创建函数复制定义时，您可以指定要复制的参数集。

- 指定 **send standby all parameters**（或省略 **all parameters** 子句）可以将存储过程的所有参数复制到备用数据库中。
- 指定 **send standby replication definition parameters** 可以只将复制定义的列复制到备用数据库中。

如果复制的存储过程没有函数复制定义，那么在执行存储过程时，Replication Server 会将其所有参数从活动数据库复制到备用数据库。每个复制存储过程只能创建一个函数复制定义。

有关在 **create applied function replication definition** 和 **create request function replication definition** 命令中使用 **send standby** 子句的详细信息，请参见《Replication Server 参考手册》中的“Replication Server 命令”。

使用复制定义来复制冗余更新

如果要复制冗余更新，则可为该列创建包含 **send standby replication definition** 参数选项的复制定义。

如果没有复制定义，Replication Server 不会将冗余更新复制到热备份中。也就是说，如果更新操作只是将当前值更改为相同的值，那么操作前映像和操作后映像完全相同，此时 Replication Server 不复制更新。

如果要复制冗余更新，请包含 **send standby replication definition** 参数选项。

如果您为表创建复制定义，Replication Server 总是 会发送冗余更新，即使该复制定义是使用 **replicate minimal columns** 选项创建的。

在热备份应用程序中使用预订

可以在热备份应用程序中使用预订。

虽然从活动数据库到备用数据库的复制操作中不使用预订，但您可以：

- 创建对逻辑主数据库中数据的预订，或者
- 通过创建预订来将其它数据库中的数据复制到逻辑复制数据库中。

create subscription 和 **define subscription** 命令使用逻辑数据库名和数据服务器名，而不使用物理名。

有关预订和预订实现的详细信息，请参见《Replication Server 管理指南第一卷》中的“管理预订”。

另请参见

- 使用复制的热备份应用程序 （第 90 页）

使用预订的限制

为了从热备份数据库复制数据或将数据复制到热备份数据库而创建预订时，存在一些限制。

Replication Server 支持在热备份应用程序中所有形式的预订实现和取消实现。适用于预订创建的限制包括：

- 如果存在数据库的逻辑连接，则不能为物理活动数据库或物理备用数据库创建预订。为了将预订数据复制到活动数据库和备用数据库，或者从这两个数据库复制数据，必须为逻辑数据库创建预订。
- 在给复制系统添加备用数据库时，不能创建预订。您必须等到备用数据库已经正确初始化之后才能创建。
- 正在创建预订时，不能给复制系统添加备用数据库。
- 正在执行 **switch active** 命令时，不能创建新预订。

逻辑主数据库的预订实现

了解逻辑主数据库的预订实现问题以及在预订实现期间为逻辑主数据库执行 **switch active** 命令时发生的情况。

在预订实现过程中，数据从活动的主数据库选入实现队列。

在执行 **switch active** 命令时，主 Replication Server 复制 RSSD 信息，以通知复制节点活动数据库已被更改。当带有实现预订的复制 Replication Server 收到此信息时，即会删除实现队列。一个新队列会通过从新的活动主数据库重新选择预订数据来构建。

注意： 必须为复制 Replication Server 运行主 Replication Server 的 RSSD 的 Replication Agent，以检测活动数据库是否已经更改。

逻辑复制数据库的预订实现

了解逻辑复制数据库的预订实现问题以及在预订实现期间为逻辑复制数据库执行 **switch active** 命令时发生的情况。

原子实现

使用原子实现时，Replication Server 会将实现队列的保存间隔设置为 **'strict'**。

在将数据应用于活动数据库并复制到备用数据库之前，事务不会从实现队列中被删除。

在应用实现队列之后，Replication Server 执行活动复制数据库中的标记。该标记表示应用实现队列之后所执行事务的开始。

在活动复制数据库中执行标记时，Replication Server 在其日志中写入如下的信息性消息：

```
I. 95/10/03 18:00:15. REPLICATE RS: Created atomic subscription
publishers_sub for replication definition publishers_rep at active
replicate
for <LDS.pubs2>
```

当标记到达备份复制数据库时，Replication Server 在其日志中写入如下的信息性消息：

```
I. 95/10/03 18:00:15. REPLICATE RS: Created atomic subscription
publishers_sub for replication definition publishers_rep at standby
replicate for <LDS.pubs2>
```

现在，实现已经完成，Replication Server 将删除实现队列。活动复制数据库和备份复制数据库中的预订均视为 VALID（有效）。

如果在正在处理实现队列时执行 **switch active** 命令，则 Replication Server 会将实现队列重新应用于新的活动数据库。如果是使用 **incrementally** 选项创建的预订，则只重新执行尚未复制到新的活动数据库的实现行的批处理。

非原子实现
如果使用非原子实现，保存间隔将设置为 0，这样，Replication Server 就能在实现队列中的行应用于活动数据库之后删除它们。

如果在执行 **switch active** 命令时正在实现预订，Replication Server 将完成对实现队列的处理，但会将预订标记为“可疑”。可以使用 **check subscription** 命令查看活动数据库和复制数据库中的预订状态。您必须删除可疑预订并重新创建预订。

注意： 异构热备份应用程序中不支持非原子实现。请参见《Replication Server 异构复制指南》的“实现”。

批量实现
如果使用批量实现创建将数据复制到热备份应用程序的预订，必须确保将预订数据装载到活动复制数据库和备份复制数据库中。

如果用记录插入行的方法（比如记录的 **bcp**）加载数据，则 Replication Server 会将这些行复制到备用数据库中。如果使用不记日志的方法装载数据，则必须将数据装载到备用数据库中，因为活动数据库日志不包含复制到备用数据库的插入记录。

在批量实现期间，在将预订数据加载到复制数据库中之前，应当先执行 **activate subscription with suspension** 命令。缺省情况下，**activate subscription with suspension** 会挂起活动数据库和备用数据库的 DSI 线程。如果挂起 DSI 线程，您就可以将数据装载到两个数据库中。

如果使用记录的 **bcp** 或记录这些行的其它某些方法来加载数据，则应执行 **activate subscription with suspension at active replicate only**，以便 Replication Server 仅挂起活动数据库的 DSI。这样，您就能将插入的行从活动数据库复制到备用数据库。

检查预订
对于逻辑复制数据库的热备份应用程序，可以使用 **check subscription** 命令检查预订状态。

根据活动数据库和备用数据库的状态是否相同，管理热备份应用程序的 Replication Server 将返回一条或两条状态消息。

例如，在创建预订时，活动数据库中的实现状态可能为 VALID（有效），备用数据库中的实现状态可能为 ACTIVATING（激活）。

删除预订
对于逻辑复制数据库，您可以使用带 **with purge** 选项的 **drop subscription** 命令删除预订。

删除预订标记跟随从 DSI 队列到活动数据库的取消实现数据，然后到达备用数据库。当两个数据库都收到该标记后，即从这两个数据库删除预订数据。

执行 switch active 时

在使用 **drop subscription** 命令和 **with purge** 选项删除预订时，可以在复制 Replication Server 上执行 **switch active** 命令。

Replication Server 挂起 DSI 线程并暂时挂起取消实现。**switch active** 完成后，DSI 线程恢复工作，取消实现也将重新开始。

可疑的 drop subscription

在以下情况下，如果使用 **with purge** 选项删除逻辑复制数据库的预订，则可能产生可疑的 **drop subscription**：

- 活动数据库中正在实现预订，并且
- 您切换活动数据库和备用数据库，然后
- 正在新的活动数据库中实现预订时，您又将预订删除。

通常，对于新活动数据库，取消实现操作会重新启动并继续，但新备用（旧活动）数据库可能保留未清除的某些预订数据。要解决这一差异，可以使用 **rs_subcmp** 程序调和活动数据库和备用数据库，也可以删除并重新创建备用数据库。

例如，尝试执行 **drop subscription** 时，可能出现下面的警告消息：

```
W. 95/10/02 20:59:15. WARNING #28171 DSI(111 SYDNEY_DS.pubs2) - /
sub_dsi.c(1231)
 REPLICATE RS: Dropped subscription publishers_sub for replication
 definition publishers_rep at standby replicate for
<SYDNEY_DS.pubs2> before
 it completed materialization at the Active Replicate. Standby
replicate may
 have some subscription data rows left in the database
```

创建备用数据库时丢失列

为具有复制定义的现有数据库创建备用数据库时，如果同时发生某些情况，将导致丢失列。

如果同时发生以下情况，可能会导致丢失列：

- 如果现有数据库的复制定义中不包括表中的所有列，并且
- 尚未提交的插入事务或更新事务在入站队列中，并且
- 为现有数据库（现在是活动数据库）创建备用数据库，随后
- 提交事务。

缺省情况下，虽然假定备用数据库接收所有列，但在事务开始时，备用数据库并不存在。Replication Server 应该已经放弃了复制定义中没有的列的值。如果某列不在复制定义中，并且备用数据库允许该列包含空值，则可以在备用数据库中插入或更新行，而不会丢失值。否则，您必须自行调和数据库。

丢失检测和恢复

创建热备份应用程序将给复制系统带来其它类型的丢失检测消息。

如果在参与热备份应用程序的 Replication Server 中重建队列，Replication Server 可能会检测到以下数据库之间的丢失：

表 13. 热备份应用程序中的丢失检测

检测到丢失的源数据库	执行的操作
逻辑复制数据库	逻辑主数据库
逻辑主数据库	物理复制数据库
物理主数据库	逻辑复制数据库
物理活动数据库	物理备用数据库
逻辑主数据库	Replication Server

要在涉及热备份应用程序的数据库恢复操作中使用 **ignore loss** 命令，请使用在您收到的丢失检测消息中出现的相同逻辑或物理数据服务器和数据库。

另请参见

- 复制系统恢复 （第 283 页）

性能调优

为满足 Replication Server 系统的需要和要求，您必须有效地管理资源并优化各个 Replication Server 的性能。

通过更改配置参数的值，通过使用并行 DSI 线程，或者，通过选择磁盘分配，可以影响 Replication Server 的性能。若要成功地管理这些资源，您应该对 Replication Server 的内部处理过程有所了解。

Replication Server 内部处理

在复制过程中，数据操作由几个 Replication Server *线程* 执行。

在 UNIX 平台上，它们是 POSIX 线程。在 Windows 平台上，它们是 WIN32 线程。Replication Server 还会将数据存储在队列中，并依赖 Replication Server 系统数据库 (RSSD) 来获得关键的系统信息。这些内部操作支持主 Replication Server 和复制 Replication Server 内的各个进程。

线程、模块和守护程序

了解线程、模块和守护程序如何在 Replication Server 中工作。

Replication Server 可以并发运行多个线程。线程总数取决于 Replication Server 管理的数据库的数量以及与该 Replication Server 有直接路由的 Replication Server 的数量。每个线程执行一项特定的功能，例如，管理用户会话、从 RepAgent 接收消息、从另一个 Replication Server 接收消息，或将事务应用于数据库。

有些线程会调用 Replication Server 的特定部分（也就是“*模块*”）来确定消息和事务的目标，并确定要复制哪些操作以及如何复制它们。

守护程序线程 在实现预订这样的 Replication Server 活动期间运行，它会在后台运行，并按预定时间或为响应某些事件而执行指定的操作。

在排除复制系统故障时，请验证 Replication Server 线程、模块和守护程序的状态。

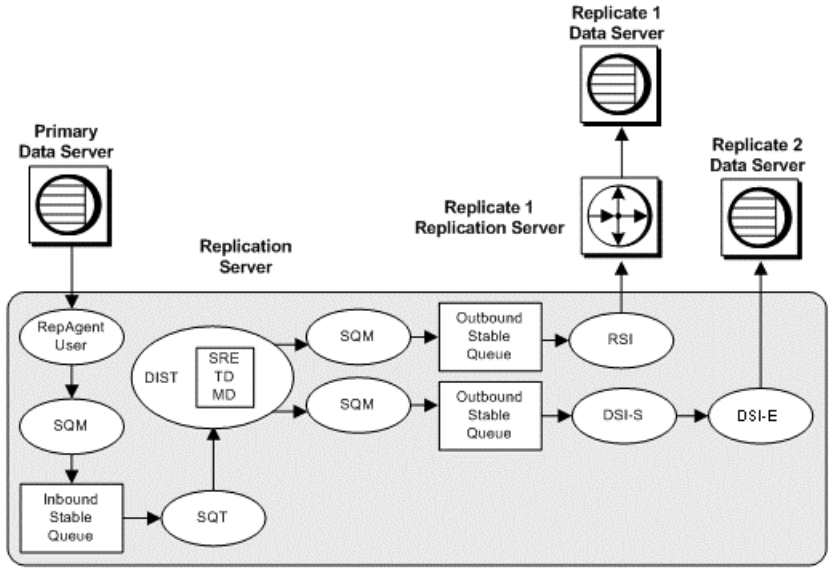
另请参见

- 主 Replication Server 中的进程 （第 106 页）
- 验证和监控 Replication Server （第 3 页）

主 Replication Server 中的进程

了解如何将源自主数据服务器的事务发送到主 Replication Server，并随后分配给复制 Replication Server。

图 11：用于主 Replication Server 中的处理的线程



Replication Agent 用户线程

了解 RepAgent 和其它 Replication Agent 如何与 Replication Server 一起使用，以便将事务信息分配给进行预订的复制数据库。

RepAgent 通过 Open Client™ 接口登录到 Replication Server。它扫描事务日志，将日志记录直接转换为 LTL（日志传输语言）命令，并在记录后立即将这些命令发送到 Replication Server（成批发送或一次发送一个命令）。之后，Replication Server 将事务信息分配给预订复制数据库。

Replication Server 为其管理的每个主数据库均提供一个 RepAgent 用户线程。因此 Replication Server 为每个 RepAgent 提供一个 RepAgent 用户线程。RepAgent 用户线程验证 RepAgent 提交的数据是否有效，并将它们写入数据库的入站稳定队列。

稳定队列管理器线程

对于主 Replication Server 访问的每个稳定队列（无论是入站队列还是出站队列），都有一个稳定队列管理器 (SQM) 线程。每个 RepAgent 用户线程与一个专用 SQM 线程协作，该 SQM 线程将在事务转发到数据服务器或其它 Replication Server 之后回收稳定队列空间。

稳定队列事务线程

稳定队列事务 (SQT) 线程重新组合事务并按提交顺序排列事务。

事务日志记录和入站队列中存储的命令会根据其提交顺序排序（尽管不一定按事务进行分组）。对于目标数据服务器上的最终应用程序以及实现处理，事务必须按提交顺序排列。

SQT 线程在从其稳定入站队列读取命令时重新组合事务，并保留事务的链接列表。对于出站队列，DSI/S 线程会安排事务，并且执行组合事务和对事务进行排序的 SQT 函数。SQT 在读取提交记录时会使该事务可供分配器 (DIST) 线程或 DSI 线程使用，具体取决于哪个进程要求对该事务进行 SQT 排序。

读取回退记录时，SQT 线程将指示 SQM 线程从所有稳定队列中删除受影响的记录。如果事务超过最大事务阈值，SQT 库将通知 DSI（由 DSI/S 线程操作）。

另请参见

- 并行 DSI 线程 （第 148 页）

分配器线程及相关模块

对于 Replication Server 管理的每个主数据库，都有一个分配器 (DIST) 线程，而该线程又使用 SQT 从入站队列读取事务，并使用 SQM 线程将事务写入出站队列。

因此，以存在三个主数据库为例，则有三个入站队列、三个 DIST 线程和三个 SQT 线程。

注意： 如果仅事务的目标是备用数据库，则会禁用 DIST 线程，且还会禁用 SQT 线程。SQM 线程存在并负责写入队列。

在确定每个事务行的目标时，DIST 线程将调用以下模块：预订分析引擎 (SRE)、事务传送和消息传送。所有 DIST 线程都共享这些模块。

预订分析引擎

预订分析引擎 (SRE) 将事务行与预订匹配。

SRE 在找到匹配项时，会将目标数据库 ID 附加到每一行。它仅标记预订所需的行，因此可以将网络通信量减到最小。如果没有匹配的预订，则 DIST 线程将放弃该行数据。

对于每一行，SRE 都会确定是否会发生预订迁移。

- 如果行的列值发生改变，导致该行与预订匹配且必须添加到复制表，那么，该行就会迁入预订。
- 如果行的列值发生改变，导致该行不再与预订匹配且必须从复制表中删除，那么，该行就会迁出 预订。

SRE 在检测到预订迁移时，会确定要复制哪个操作（插入、删除或更新），以保持复制表和主表之间的一致性。

事务传送模块
事务传送 (TD) 模块由 DIST 线程调用，以打包事务行并分配给数据服务器和其它 Replication Server。

消息传送模块
消息传送 (MD) 模块由 DIST 线程调用，以优化事务到数据服务器或其它 Replication Server 的路由。

DIST 线程会将事务行和目标 ID 传递到 MD 模块。该模块会使用这些信息和 RSSD 中的路由信息来确定将事务发送到何处：

- 数据服务器（通过 DSI 线程），还是
- Replication Server（通过 RSI 线程）。

MD 模块在确定如何发送事务后，将事务置于相应的出站队列中。

分配器状态记录
Replication Server 在 RSSD 的 Replication Server `rs_databases` 系统表中记录分配器线程的 DIST 状态。

分配器 (DIST) 线程从入站队列读取事务并将复制的事务写入出站队列。Replication Server 连接到主数据库时会创建 DIST 线程，可以手动或通过 Replication Server 配置挂起或恢复该线程。恢复和挂起 DIST 线程会修改该线程的 DIST 状态。

`rs_databases` 中的记录允许 DIST 线程即使在 Replication Server 关闭后也保留其状态。

请参见《Replication Server 参考手册》的“Replication Server 系统表”的“`rs_databases`”。

数据服务器接口线程
Replication Server 启动 DSI 线程，将事务提交到与该 Replication Server 保持连接的复制数据库。

每个 DSI 线程都由一个调度程序线程 (DSI-S) 和一个或多个执行程序线程 (DSI-E) 组成。每个 DSI 执行程序线程都打开一个与数据库的 Open Client 连接。

若要改进将事务从 Replication Server 发送到它管理的复制数据库的性能，您可以配置数据库连接，以便使用多个 DSI 执行程序线程来应用事务（即，使用并行 DSI 线程）。

DSI 调度程序线程调用 SQT 接口来执行以下任务：

- 按提交顺序将小事务集中分组
- 将事务组派发给下一个可用 DSI 执行程序线程

DSI 执行程序线程：

- 根据指派给数据库连接的函数字符串类，使用为函数定义的函数字符串来映射函数。
- 在复制数据库中执行事务
- 对数据服务器返回的所有错误执行相应的操作；并在例外日志中记录所有失败事务（取决于指派的错误操作）

DSI 线程可能应用来自 Replication Server 支持的所有主数据库的多种事务。将从复制数据服务器的一个出站稳定队列中读取这些事务。

另请参见

- 并行 DSI 线程 （第 148 页）

Replication Server 接口线程

RSI 线程是将消息从一个 Replication Server 发送到另一个 Replication Server 的异步接口。对于与源数据库具有直接路由的每个目标 Replication Server，都存在一个 RSI 线程。

主 Replication Server 中的 DIST 线程处理事务，导致将发往其它 Replication Server 的事务写入 RSI 出站队列。RSI 线程登录到每个复制 Replication Server，并将消息从稳定队列传输到复制 Replication Server。

创建从一个 Replication Server 到另一个 Replication Server 的直接路由后，源 Replication Server 中的 RSI 线程会登录到复制 Replication Server。创建了间接路由后，Replication Server 不会创建新的稳定队列和 RSI 线程。而是由直接路由的 RSI 线程来处理间接路由的消息。请参见《Replication Server 管理指南第一卷》中的“管理路由”。

杂项守护程序线程

一些 Replication Server 守护程序线程可以在复制系统中执行杂项任务。

表 14. 其它 Replication Server 守护程序线程

线程或守护程序的名称	说明
报警守护程序 (dALARM)	报警守护程序跟踪其它线程设置的报警，例如，连接的淡出时间和预订重试守护程序的时间间隔。
异步 I/O 守护程序 (dAIO)	异步 I/O 守护程序管理到 Replication Server 稳定队列的异步 I/O。
连接管理器守护程序 (dCM)	连接管理器守护程序管理与数据服务器和其它 Replication Server 的连接。
恢复守护程序 (dREC)	恢复守护程序管理与热备份应用程序、路由和恢复过程相关的各种操作。
预订重试守护程序 (dSUB)	预订重试守护程序在可配置的超时周期（**rs_config** 系统表中的 `sub_daemon_sleep_time` 配置参数）之后被唤醒，并尝试重新开始处理可能已经失败的预订。

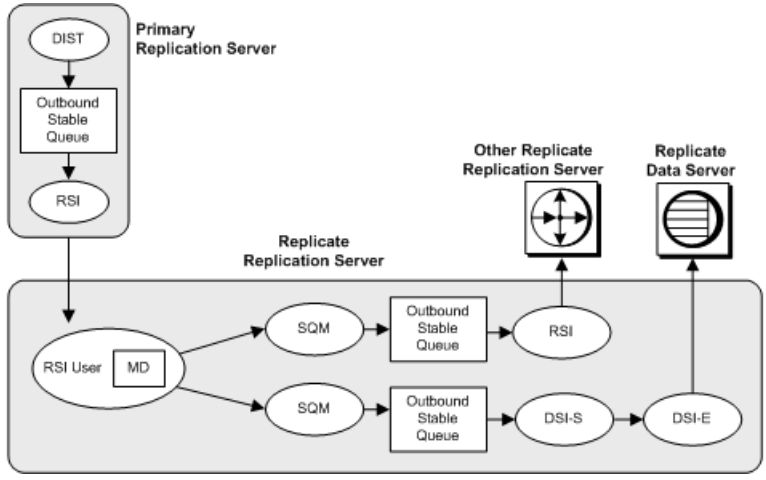
线程或守护程序的名称	说明
版本守护程序 (dVERSION)	Replication Server 在升级之后首次启动时，版本守护程序会短暂激活。它会将 Replication Server 的新版本号传送到 ID 服务器。
RS 用户线程	在创建或删除预订的过程中，RS 用户线程会管理来自复制 Replication Server 的连接。 有关创建和删除预订涉及的数据流，请参见《Replication Server 管理指南第一卷》的“管理预订”中的“预订实现方法”。
USER 线程	用户登录到 Replication Server 执行 RCL 命令时，会创建 USER 线程。

复制 Replication Server 中的处理

了解复制 Replication Server 从主 Replication Server 接收传入消息时涉及的过程。

图中显示了主 Replication Server 中的过程还涉及的一些相同的线程（SQM、RSI、DSI）。

图 12： 复制 Replication Server 中的事务处理



另请参见

- 主 Replication Server 中的进程 （第 106 页）

RSI 用户线程

RSI 用户线程是一个客户端连接线程，针对来自其它 Replication Server 的传入消息。

它会调用消息传送 (MD) 模块，以确定是否将消息发送到以下位置：

- 使用 DSI 线程的数据服务器。DSI 线程由一个调度程序线程 (DSI-S) 以及一个或多个执行程序线程 (DSI-E) 组成。
- 另一个使用 RSI 线程的 Replication Server。

RSI 用户线程将发往其它 Replication Server 或数据库的命令写入出站队列。在出站队列中存储消息后，主 Replication Server 中的过程涉及的线程将处理这些消息。

另请参见

- 数据服务器接口线程 （第 108 页）
- Replication Server 接口线程 （第 109 页）
- 主 Replication Server 中的进程 （第 106 页）

影响性能的配置参数

Replication Server 提供了用于提高性能的配置参数，这些配置参数会影响整个服务器或只影响单个连接或路由。

影响性能的 Replication Server 参数

您可以更改这些配置参数的值以提高 Replication Server 性能。

在您安装 Replication Server 之后，**rs_init** 会设置缺省配置参数。

有关如何使用 **configure replication server** 修改这些参数的信息，请参见《Replication Server 管理指南第一卷》的“管理复制系统”的“设置 Replication Server 配置参数”中的“更改 Replication Server 参数”。

表 15. 影响性能的 Replication Server 参数

配置参数	说明
block_size to '*value*' **with shutdown**	指定队列块大小，它是稳定队列结构使用的连续内存块中的字节数。 有效值：16KB、32KB、64KB、128KB 或 256KB 缺省值：16KB **注意：** 在执行此命令以更改块大小时，Replication Server 将关闭。在 Replication Server 15.6 之前的版本中，在指定块大小后，必须包括 **with shutdown** 子句。在 15.6 和更高版本中，**with shutdown** 子句是可选的；您无需重新启动 Replication Server 即可使队列块大小更改生效。只应使用 **configure replication server** 命令更改此参数。否则，可能会损坏队列。 许可证：在高级服务选件下单独许可。

配置参数	说明
db_packet_size	网络包的最大大小。数据库通信期间，网络包的值必须在数据库所接受的范围内。如果已经重新配置 Adaptive Server，则可以更改此值。 最大值：16,384 字节 缺省值：对于所有 Adaptive Server 数据库，网络包最大大小的缺省值为 512 字节
deferred_queue_size	Open Server 延迟队列的最大大小。如果超过 Open Server 限制，应增大此最大大小。此值必须大于 0。 **注意：** 如果进行更改，必须重新启动 Replication Server 以使更改生效。 缺省值：Linux 和 HPIA32 上为 2,048；其它平台上为 1024
disk_affinity	指定用于分配下一个分区的分配提示。输入一个分区的逻辑名称，在当前分区已满时应该将下一个段分配给此分区。值为“*partition_name*”和“off”。 缺省值：off
dist_direct_cache_read	让分配器 (DIST) 线程能直接从稳定队列事务线程 (SQT) 高速缓存中读取 SQL 语句。这减少了 SQT 的负载以及二者之间的依赖性，并提高了 SQT 和 DIST 的效率。 缺省值：off
dsi_bulk_copy	为连接打开或关闭批量拷入功能。如果将 **dynamic_sql** 和 **dsi_bulk_copy** 均设置为 on，DSI 将应用批量拷入。如果未使用批量拷入，则会使用动态 SQL。如果进行大批量插入，Sybase 建议您打开 **dsi_bulk_copy** 以提高性能。 缺省值：off。
dsi_bulk_threshold	事务中的连续 **insert** 命令数，在达到此数字时，将会触发 Replication Server 使用批量拷入。当稳定队列事务 (SQT) 遇到大批量 **insert** 命令时，它将在内存中保留指定数量的 **insert** 命令以确定是否应用批量拷入。由于这些命令保留在内存中，Sybase 建议您不要将该值配置为比 **dsi_large_xact_size** 配置值高太多。 Replication Server 使用 **dsi_bulk_threshold** 处理到 Sybase IQ 的实时装载 (RTL) 复制和到 Adaptive Server 的高容量自适应复制 (HVAR)。如果对某个表执行的 insert、delete 或 update 操作的命令数少于在编译后指定的数量，RTL 和 HVAR 将使用语言而不是批量接口。 最小值：1 缺省值：20
dsi_cmd_batch_size	Replication Server 放在批处理命令中的字节数的最大值。 缺省值：8192 字节

配置参数	说明
dsi_cmd_prefetch	允许DSI在等待数据服务器的响应时构建下一批命令，从而提高了DSI效率。如果您还调优数据服务器以增强性能，在使用此功能时，可能会进一步提高性能。 缺省值：off 在将 **dsi_compile_enable** 设置为“on”时，Replication Server 会忽略您为 **dsi_cmd_prefetch** 设置的内容。 许可证：在高级服务选件下单独许可。
dsi_max_xacts_in_group	指定组中事务的最大数量。较大的数字可以改善复制数据库的数据延迟问题。值的范围：1 – 1000. 缺省值：20 当打开 **dsi_compile_enable** 时，将忽略此参数。
dsi_non_blocking_commit	指定在提交后将 Replication Server 保存消息的时间延长的分钟数。值的范围：0 – 60 分钟。 缺省值：0 – 禁用非阻塞提交。 在使用 Adaptive Server 15.0 和更高版本的 **delayed_commit** 选项或 Oracle 10g v2 中的等效功能时，启用此参数可提高复制性能。
dsi_xact_group_size	置入一个分组事务的最大字节数，包括稳定队列开销。分组事务是DSI作为单个事务应用的一组事务。值为 –1 意味着无分组。 Sybase 建议将 **dsi_xact_group_size** 设置为最大值，并使用 **dsi_max_xacts_in_group** 控制组中的事务数。 **注意：** 此内容不适用于 Replication Server 15.0 版和更高版本。这是为了与旧版本的 Replication Server 保持兼容而保留的。 最大值：2,147,483,647 缺省值：65,536 字节 当打开 **dsi_compile_enable** 时，将忽略此参数。
dynamic_sql	打开或关闭动态 SQL 功能。只有在将此参数设置为“on”时，与动态 SQL 相关的其它配置参数才生效。 缺省值：off
dynamic_sql_cache_size	为 Replication Server 指定可使用连接的动态 SQL 的数据库对象数。 缺省值：100 最小值：1 最大值：65,536

配置参数	说明
dynamic_sql_cache_management	管理 DSI 执行程序线程的动态 SQL 高速缓存。值：**mru** - 保留最近使用的语句并释放其余语句，以便在达到 **dynamic_sql_cache_size** 时分配新的动态语句。**fixed**（缺省值）- 一旦达到 **dynamic_sql_cache_size**，Replication Server 将停止分配新的动态语句。
exec_cmds_per_timeslice	通过使用 **exec_cmds_per_timeslice** 指定 LTI 或 RepAgent 执行程序线程在放弃 CPU 之前可以处理的 LTL 命令数，控制 RepAgent 执行程序可以处理的命令数。通过增大此值，您可以使 RepAgent 执行程序线程更长时间地控制 CPU 资源，这样可以提高从 RepAgent 到 Replication Server 的吞吐量。 使用 **alter connection** 在连接级别设置此参数。 缺省值：2,147,483,647 最小值：1 最大值：2,147,483,647
exec_nrm_request_limit	指定可用于主数据库中等待规范化的消息的内存量。 在使用 **exec_nrm_request_limit** 之前，使用 **configure replication server** 将 **nrm_thread** 设置为“on”。 缺省值：1,048,576 字节 (1MB) 最小值：16,384 字节 (16KB) 最大值：2,147,483,647 字节 (2GB) 许可证：在高级服务选件下单独许可。
exec_sqm_write_request_limit	指定可用于等待写入到入站队列中的消息的内存量。 缺省值：1MB，最小值：16KB，最大值：2GB
init_sqm_write_delay	在向队列中写入部分填满的消息块之前，SQM Writer 应该等待更多消息的初始时间。SQM Writer 总是尝试向队列中写入填满的块。如果 SQM Writer 已经对某个块进行了部分填充，但无法填满它，就会等待 **init_sqm_write_delay** 所指定的时间，然后再重新检查是否有消息在等待添加到块中。如果没有消息，SQM Writer 会将 **init_sqm_write_delay** 时间加倍。SQM Writer 将不断将延迟时间加倍，直到延迟时间达到 **init_sqm_write_max_delay** 的值。此时，SQM Writer 会写入部分填满的块。 缺省值：100 毫秒
init_sqm_write_max_delay	在向队列中写入部分填满的消息块之前，SQM Writer 线程应该等待更多消息的最长时间。有关详细信息，请参见 **init_sqm_write_delay** 的说明。 缺省值：1000 毫秒

配置参数	说明
mem_reduce_malloc	启用后可以在更大的块中分配内存，这会减少内存分配次数，从而提高 Replication Server 性能。 缺省值：off 许可证：在高级服务选件下单独许可。
mem_thr_dsi	指定用于强制 DSI 线程停止填充 SQT 高速缓存的总内存百分比。 缺省值：**memory_limit** 值的 80%。 范围：1 – 100
mem_thr_exec	指定用于强制 EXEC 线程停止从 RepAgent 接收命令的总内存百分比。 缺省值：**memory_limit** 值的 90%。 范围：1 – 100
mem_thr_sqt	指定用于强制 SQT 线程刷新其高速缓存中的最大事务的总内存百分比（如果可能）。 缺省值：**memory_limit** 值的 85%。 范围：1 – 100
mem_warning_thr1	指定在生成第一条警告消息之前使用的总内存阈值百分比。请参见 **memory_limit**。 缺省值：**memory_limit** 值的 80%。 范围：1 – 100
mem_warning_thr2	指定在生成第二条警告消息之前使用的总内存阈值百分比。请参见 **memory_limit**。 缺省值：**memory_limit** 值的 90%。 范围：1 – 100
memory_control	管理需要大量内存的线程的内存控制行为。请参见 **memory_limit**。 值为： • on – 启用内存控制 • off – 禁用内存控制 缺省值：on

配置参数	说明
memory_limit	Replication Server 可以使用的最大内存总量 (MB)。 **memory_limit** 指示的内存池中的可用内存量与其它几个配置参数的值直接相关。这些配置参数包括 **exec_nrm_request_limit**、**exec_sqm_write_request_limit**、**md_sqm_write_request_limit**、**queue_dump_buffer_size**、**sqt_max_cache_size**、**sre_reserve** 和 **sts_cachesize**。 缺省值：2,047 对于 32 位 Replication Server： • 最小值 – 0 • 最大值 – 2,047 对于 64 位 Replication Server： • 最小值 – 0 • 最大值 – 2,147,483,647 如果 **memory_control** 为： • on – 当内存消耗量超过 **memory_limit** 时，Replication Server 不关闭 • off – 当内存消耗量超过 **memory_limit** 时，Replication Server 自动关闭 监控内存使用情况并在需要时增大 **memory_limit**。 在 Replication Server 中，需要大量内存的线程是： • EXEC • SQT • DST 这些线程在接收或处理新数据之前通过执行内存使用量检查来执行内存控制。在内存控制期间，如果内存使用量很高，则会通过以下措施来调整线程运行情况： • 阻止线程对新数据进行分组，并清除和处理现有数据；或者， • 使线程进入休眠状态，以使它在内存可用之前不接收新数据。 如果您设置的值大于 2,047，则降级会将该值重置为 2,047 以防止溢出。

配置参数	说明
md_sqm_write_request_limit	指定分配器可用于存放等待写入出站队列的消息的内存量。 **注意：** 在 Replication Server 12.1 中，**md_sqm_write_request_limit** 将替换 **md_source_memory_pool**。将保留 **md_source_memory_pool** 以便与旧版本的 Replication Server 保持兼容。 缺省值：1MB 最小值：16K 最大值：2GB
nrm_thread	启用 NRM 线程，Replication Server 可使用 NRM 线程在由 RepAgent 执行程序线程进行分析的同时并行规范化并打包日志传送语言 (LTL) 命令。NRM 线程进行的并行处理减少了 RepAgent 执行程序线程的响应时间。NRM 线程是从 RepAgent 执行程序线程拆分的线程。 在使用 **exec_nrm_request_limit** 之前使用 **configure replication server** 命令将 **nrm_thread** 设置为 on。 缺省值：off 许可证：在高级服务选件下单独许可。
rec_daemon_sleep_time	指定恢复守护程序的休眠时间以设置唤醒时间间隔，该守护程序在热备份应用和某些其它操作中处理“strict”保存间隔消息。 缺省值：2 分钟
smp_enable	启用对称多重处理 (SMP)。指定 Replication Server 线程应由 Replication Server 内部预定，还是由操作系统外部预定。当 Replication Server 线程由内部预定时，无论有多少个可用处理器，都会将 Replication Server 限定为一个计算机处理器。值为“on”和“off”。 缺省值：on 升级或降级不会更改您已设置的值。
sqm_async_seg_delete	将 **sqm_async_seg_delete** 设置为 on 可启用用于删除段的专用守护程序并改进入站和出站队列处理的性能。 缺省值：on 必须重新启动 Replication Server 以使参数设置的任何更改生效。 如果 **sqm_async_seg_delete** 为 on，则 Replication Server 可能需要较大的分区。使用 **alter partition** 扩展分区。请参见： • 《Replication Server 配置指南》的“准备安装和配置 Replication Server”的“规划复制系统”中的“每个 Replication Server 的初始磁盘分区”。 • 《Replication Server 管理指南第一卷》的“Replication Server 技术概述”的“使用 Replication Server 处理事务”的“稳定队列”中的“稳定队列的分区”。

配置参数	说明
sqm_cache_enable	指示是否在稳定设备中启用 SQM 高速缓存和大 I/O。 缺省值：on
sqm_cache_size	指示高速缓存中的页数，其中页大小由 **sqm_page_size** 指定。范围是从 1 到 4096。 缺省值：16
sqm_page_size	指示页中的块数量。 以每页包含多少个块的形式设置服务器端稳定队列页大小。将页大小用单引号或双引号引起来。例如，将页大小设置为 4 会指示 Replication Server 以 64K 大块的形式向稳定队列中写入数据。 配置页大小也会设置 Replication Server 的 I/O 大小。范围是 1 到 64。 缺省值：4
sqm_recover_segs	指定 Replication Server 在使用恢复 QID 信息更新 RSSD 之前分配的稳定队列段数。 Sybase 建议您增大 **sqm_recover_segs** 的值以提高性能。 缺省值：1 最小值：1 最大值：2,147,483,648
sqm_write_flush	**sqm_write_flush** 指定在写操作完成之前是否将内存缓冲区中的写入内容刷新到磁盘（稳定设备注意事项）。值为“on”、“off”和“dio”。 缺省值：on
sqt_init_read_delay	SQT 线程在检查其命令队列中是否有新指令之前等待 SQM 读取时休眠的时间长度。每当休眠时间到期时，如果此命令队列为空，SQT 会使其休眠时间加倍，最大可达到 **sqt_max_read_delay** 设置的值。 缺省值：1 毫秒 (ms) 最小值：0 毫秒 最大值：86,400,000 毫秒（24 小时）

配置参数	说明
sqt_max_cache_size	可以使用 **sqt_max_cache_size** 将 SQT 高速缓存大小调整为 SQT 高速缓存最大值（以字节为单位）。 对于 32 位 Replication Server： • 缺省值 – 1,048,576 • 最小值 – 0 • 最大值 – 2,147,483,647 对于 64 位 Replication Server： • 缺省值 – 20,971,520 • 最小值 – 0 • 最大值 – 2,251,799,813,685,247 如果设置的值大于 2,147,483,647 字节，则降级会将该值重置为 2,147,483,647 字节以防止溢出。
sqt_max_prs_size	由 HVAR 和 RTL 的事务分析进程解包的命令消耗的最大内存（以字节为单位）。 对于 32 位 Replication Server： • 缺省值 – 2,147,483,647 (2GB) • 最小值 – 0 • 最大值 – 2,147,483,647 对于 64 位 Replication Server： • 缺省值 – 2,147,483,647 (2GB) • 最小值 – 0 • 最大值 – 2,251,799,813,685,247 在服务器级别使用 **configure replication server** 为所有连接设置该参数，或者在数据库级使用 **alter connection** 为单独的连接设置该参数。数据库级别的缺省值为 0。如果您保留数据库级缺省值或重置为缺省值，Replication Server 将使用在服务器级别设置的值。 **注意：** 在升级到 Replication Server 15.7.1 和更高版本后，必须对 32 位和 64 位 Replication Server 将缺省值设置为 2GB。
sqt_max_read_delay	SQT 线程在检查其命令队列中是否有新指令之前等待 SQM 读取时休眠的最大时间长度。 缺省值：1 毫秒 最小值：0 毫秒 最大值：86,400,000 毫秒（24 小时）

配置参数	说明
sts_cachesize	要高速缓存系统表，请使用 **sts_cachesize** 指定为每个高速缓存的 RSSD 系统表高速缓存的总行数。将此数字增大到活动复制定义的数目，可防止 Replication Server 执行占用资源较多的表查询。 通过检查计数器 11008 – STSCacheExceed 或检查 Replication Server 日志中是否有已从 STS 高速缓存中删除行的警告，来监控 STS 高速缓存是否太小。 缺省值：1000
sts_full_cache_*system_table_name*	要高速缓存系统表，请使用 **sts_full_cache**_*system_table_name* 指定要完全高速缓存的 RSSD 系统表。对于简单的 **select** 语句，完全高速缓存的表不需要访问 RSSD。只能完全高速缓存某些 RSSD 表。 缺省值：`rs_asyncfuncs`、`rs_clsfunctionsrs_columns`、`rs_objects`、`rs_objfunctions`、`rs_repobjs` 和 `rs_users` 是完全高速缓存的。Sybase 建议您高速缓存这些表以提高性能。
sub_daemon_sleep_time	要设置唤醒时间间隔，请使用 **sub_daemon_sleep_time** 设置预订守护程序在醒来恢复预订前的休眠时间（以秒为单位）。范围是 1 到 31,536,000。 缺省值：120 秒
sub_sqm_write_request_limit	指定预订实现线程或取消实现线程用于存放等待写入出站队列的消息的内存。 缺省值：1MB 最小值：16K 最大值：2GB

另请参见

- 增加队列块大小 （第 212 页）
- 高级服务选件 （第 196 页）
- 设置 SQM Writer 等待的时间 （第 131 页）
- 设置唤醒时间间隔 （第 141 页）
- 稳定设备：注意事项 （第 121 页）
- 设置 SQT 高速缓存大小 （第 142 页）
- 高速缓存系统表 （第 131 页）
- 控制 RepAgent 执行程序可以处理的命令数 （第 143 页）
- 使 SMP 更有效 （第 144 页）
- 指定分配的稳定队列段数 （第 144 页）

稳定设备：注意事项

与任何应用程序一样，Replication Server 也采用标准的 I/O 和 I/O 设备最佳操作。在计划如何使用稳定设备来支持稳定队列时，您应该考虑对磁盘读/写磁头和 I/O 通道的争用产生的影响。

在达到可以将一个或多个设备专用于每个队列的程度时，I/O 不大可能产生性能问题。这包括防止其它进程（如主数据库、复制数据库或 RSSD）使用设备。您可以使用数据库连接参数 **disk_affinity** 在专用设备支持的队列和特定分区之间建立关系。

对于在 UNIX 操作系统文件上初始化的稳定队列，**sqm_write_flush** 配置参数控制在写操作完成之前是否将内存缓冲区中的写入内容刷新到磁盘。

当 **sqm_write_flush** 为 on 时，Replication Server 会使用 O_DSYNC 标志打开稳定队列。此标志用于确保在写操作完成之前将写入内容从内存缓冲区刷新到磁盘。由于数据存储在物理介质上，因此 Replication Server 始终可以在出现系统故障时恢复数据。这是缺省设置。

当 **sqm_write_flush** 为 off 时，在 UNIX 文件系统中可能会缓冲写入。如果后续写入失败，则无法保证自动恢复。测试已经表明：在比较分区类型和 I/O 刷新的各种选项的写入速率时，在 **sqm_write_flush** 为“on”的情况下写入已缓冲文件系统的速率仅仅是写入原始分区的速率的五分之一。

此外，测试还表明：写入原始分区的速率仅仅是在 **sqm_write_flush** 为“off”的情况下写入已缓冲文件系统的速率的七分之一。如果在对稳定设备使用“UNIX 缓冲”文件系统时关闭 **sqm_write_flush**，可提供峰值 I/O 性能，但会增加数据丢失的风险。假如保留主数据库的事务日志备份，则可以减小或消除该风险。

对于文件系统分区而言，与同步 I/O DSYNC 相比，直接 I/O 可减少 I/O 延迟。使用以下命令配置直接 I/O：

```
configure replication server set sqm_write_flush to
"dio"
```

此命令可启用直接 I/O 并仅在稳定队列位于文件系统中时才有效。直接 I/O 方法允许 Replication Server 直接读取或写入磁盘，而不必缓冲文件系统。正确调整稳定队列高速缓存。适当的高速缓存大小可确保大多数读取事务能够在高速缓存内完成。

注意： 对于 Replication Server 15.1 和更高版本，仅在 Solaris 和 Linux 平台上支持直接 I/O。

此命令是静态的，也就是，必须重新启动服务器，才能使其生效。

注意： 对于在原始分区或 Windows 文件上初始化的稳定队列，忽略 **sqm_write_flush** 设置。在这些情况下，始终直接写入介质。

为提高 I/O 性能，Replication Server 15.1 和更高版本支持稳定设备的高速缓存。

另请参见

- 稳定队列高速缓存 （第 134 页）

影响性能的连接参数

Replication Server 提供了一些可能影响性能的数据库连接参数。

有关连接参数的完整列表，请参见《Replication Server 管理指南第一卷》中的“管理数据库连接”。

表 16. 影响性能的连接参数

配置参数	说明
async_parser	使 Replication Server 可以异步分析来自 RepAgent 的命令。 如果将 **async_parser** 设置为 on，则会： • 启用异步分析程序 – 将 **exec_prs_num_threads** 设置为 2 • 启用 ASCII 打包 – 将 **ascii_pack_ibq** 设置为 on • 启用入站直接命令复制 – 将 **cmd_direct_replicate** 设置为 on • 启用出站直接命令复制 – 将 **dist_cmd_direct_replicate** 设置为 on 缺省值：off **注意：** 在配置异步分析程序之前，请确保 **smp_enable** 设置为 on，并且 Replication Server 主机可以支持使用额外的线程进行分析。必须将 Replication Server 节点版本设置为 1571 或更高版本，然后才能将 **ascii_pack_ibq** 设置为 on。如果节点版本早于 1571，将 **async_parser** 设置为 on 只会设置 **exec_prs_num_threads**、**cmd_direct_replicate** 和 **dist_cmd_direct_replicate**。
ascii_pack_ibq	通过使用 ASCII 打包，减少入站队列中的打包命令所消耗的稳定队列存储空间。 缺省值：off **注意：** 您必须为 Replication Server 启用异步语法分析程序才能在入站队列中从 ASCII 包受益。必须将 Replication Server 节点版本设置为 1571 或更高版本，然后才能将 **ascii_pack_ibq** 设置为 on。
batch	缺省值“on”允许将命令批处理发送到复制数据库。 缺省值：on
cmd_direct_replicate	针对执行程序线程将 **cmd_direct_replicate** 设置为 on，以便将分析的数据连同二进制或 ASCII 数据一起直接发送到执行程序线程。当需要时，分配器线程可以直接从分析的数据中检索数据并进行处理，通过节省重新分析数据所花费的时间来提高复制性能。 缺省值：off

配置参数	说明
dist_cmd_direct_replicate	将 **dist_cmd_direct_replicate** 设置为 on 可允许 DIST 模块通过内存高速缓存发送内部分析的数据。 缺省值：on 如果将 **dist_cmd_direct_replicate** 设置为 off，DIST 模块将通过出站队列将数据发送到 DSI。
db_packet_size	网络包的最大大小。数据库通信期间，网络包的值必须在数据库所接受的范围内。 最大值：16384 字节 缺省值：对于所有 Adaptive Server 数据库，网络包最大大小的缺省值为 512 字节
disk_affinity	指定用于分配下一个分区的分配提示。输入一个分区的逻辑名称，在当前分区已满时应该将下一个段分配给此分区。值为“*partition_name*”和“off”。 缺省值：off
dist_sqt_max_cache_size	入站队列的最大稳定队列事务 (SQT) 高速缓存大小（以字节为单位）。缺省值为 0，表示 **sqt_max_cache_size** 参数的当前设置用作连接的高速缓存最大大小。 缺省值：0 对于 32 位 Replication Server： • 最小值 – 0 • 最大值 – 2,147,483,647 对于 64 位 Replication Server： • 最小值 – 0 • 最大值 – 2,251,799,813,685,247

配置参数	说明
dsi_cdb_max_size	Replication Server 可为 HVAR 或 RTL 生成的最大净更改数据库大小（以兆字节为单位）。 • 缺省值 – 1024 • 最小值 – 0 • 最大值 – 2,147,483,647 在 HVAR 中，Replication Server 使用 **dsi_cdb_max_size** 作为阈值以便： • 检测随后使用连续复制模式复制的大事务。 • 停止将小型可编译事务分组到需要大于 **dsi_cdb_max_size** 的净更改数据库的组中。 在 RTL 中，Replication Server 使用 **dsi_cdb_max_size** 通过完全增量编译以增量方式刷新大事务组。
dsi_cmd_batch_size	Replication Server 放在批处理命令中的字节数的最大值。 缺省值：8192 字节
dsi_cmd_prefetch	允许 DSI 在等待数据服务器的响应时预先构建下一批命令，因此可提高 DSI 效率。如果您还调优数据服务器以增强性能，则在使用此功能时，您可能会获得额外的性能提高。 缺省值：off 在将 **dsi_compile_enable** 设置为“on”时，Replication Server 会忽略您为 **dsi_cmd_prefetch** 设置的内容。 许可证：在高级服务选件下单独许可。
dsi_commit_check_locks_intrvl	DSI 执行程序线程在 `rs_dsi_check_thread_lock` 函数字符串的执行之间等待的毫秒数 (ms)。用于并行 DSI。 缺省值：1000 毫秒（1 秒） 最小值：0 最大值：86,400,000 毫秒（24 小时）
dsi_commit_check_locks_max	在回退并重试事务之前，DSI 执行程序线程执行 `rs_dsi_check_thread_lock` 函数字符串的最多次数。用于并行 DSI。 缺省值：400 最小值：1 最大值：1,000,000

配置参数	说明
dsi_commit_control	指定提交控制处理是由 Replication Server 使用内部表在内部进行处理 (on)，还是使用 `rs_threads` 系统表在外部进行处理 (off)。用于并行 DSI。 缺省值：on
dsi_isolation_level	指定事务的隔离级别。ANSI 标准和 Adaptive Server 支持的值包括： • 0 – 确保一个事务所写入的数据表示实际数据。 • 1 – 防止脏读，并确保一个事务所写入的数据表示实际数据。 • 2 – 防止非重复读取和脏读，并确保一个事务所写入的数据表示实际数据。 • 3 – 防止幻像行、非重复读取和脏读，并确保一个事务所写入的数据表示实际数据。 通过使用自定义函数字符串，Replication Server 可以支持复制数据服务器可使用的任何隔离级别。不再仅限于支持 ANSI 标准。 缺省值：目标数据服务器的当前事务隔离级别
dsi_large_xact_size	事务中允许包含的命令的数量，一旦超过此数值，此事务将被视为大事务。 缺省值：100 最小值：4 最大值：2,147,483,647 当打开 **dsi_compile_enable** 时，将忽略此参数。
dsi_max_cmds_in_batch	定义可对其输出命令进行批处理的源命令的最大数。 必须挂起并恢复连接，参数中的任何更改才能生效。 范围：1 – 1000 缺省值：100
dsi_max_xacts_in_group	指定组中事务的最大数量。较大的数字可以改善复制数据库的数据延迟问题。值的范围：1 – 1000. 缺省值：20 当打开 **dsi_compile_enable** 时，将忽略此参数。
dsi_num_large_xact_threads	为用于大事务而保留的并行 DSI 线程的数量。最大值是 **dsi_num_threads** 的值减去 1。 缺省值：0
dsi_num_threads	要使用的并行 DSI 线程的数量。最大值为 255。 缺省值：1

配置参数	说明
dsi_partitioning_rule	指定 DSI 用来将可用的并行 DSI 线程中的事务分区的分区规则（一个或多个）。值为 **origin**、**origin_sessid**、**time**、**user**、**name** 和 **none**。 缺省值：none 当打开 **dsi_compile_enable** 时，将忽略此参数。
dsi_serialization_method	指定用于确定何时可以启动事务而又仍保持一致性的方法。在任何情况下，都会保留提交顺序。 这些方法按并行度的大小从高到低进行排序。并行度越大，将会导致在将并行事务应用于复制数据库时，这些并行事务之间的争用越多。若要减少争用，请使用 **dsi_partitioning_rule** 选项。 • **no_wait** – 指定事务一旦就绪即可启动，而不考虑其它事务的状态。 **注意：** 如果 **dsi_commit_control** 设置为“on”，则只能将 **dsi_serialization_method** 设置为 **no_wait**。 • **wait_for_start** – 指定某个事务可以在预定在它紧前面提交的事务启动后立即启动。 • **wait_for_commit**（缺省值） – 指定事务直到安排在其紧前面提交的事务准备好提交后才可开始。 • **wait_after_commit** – 指定一个事务要等到其前一个预定提交的事务已完全提交后才启动。 保留下面这些选项，仅仅是为了向后兼容早期版本的 Replication Server： • **none** – 与 **wait_for_start** 相同。 • **single_transaction_per_origin** – 与 **dsi_partitioning_rule** 设置为 **origin** 的 **wait_for_start** 相同。 **注意：** 不再支持将 **isolation_level_3** 值作为序列化方法，但它与将 **dsi_serialization_method** 设置为 **wait_for_start** 并将 **dsi_isolation_level** 设置为 3 的操作相同。 缺省值：**wait_for_commit**

配置参数	说明
dsi_sqt_max_cache_size	出站队列的 SQT（稳定队列事务）接口高速缓存大小的最大值（以字节为单位）。缺省值为 0，表示 **sqt_max_cache_size** 参数的当前设置用作连接的高速缓存最大大小。 缺省值：0 对于 32 位 Replication Server： • 最小值 – 0 • 最大值 – 2GB（2,147,483,648 字节） 对于 64 位 Replication Server： • 最小值 – 0 • 最大值 – 2 PB（2,251,799,813,685,247 字节）
dsi_xact_group_size	置入一个分组事务的最大字节数，包括稳定队列开销。分组事务是 DSI 作为单个事务应用的一组事务。值为 –1 意味着无分组。 Sybase 建议将 **dsi_xact_group_size** 设置为最大值，并使用 **dsi_max_xacts_in_group** 控制组中的事务数。 **注意：** 此内容不适用于 Replication Server 15.0 版和更高版本。这是为了与旧版本的 Replication Server 保持兼容而保留的。 最大值：2,147,483,647 缺省值：65,536 字节 当打开 **dsi_compile_enable** 时，将忽略此参数。
exec_cmds_per_time-slice	指定 LTI 或 RepAgent 执行程序线程在放弃 CPU 之前可以处理的 LTL 命令数。通过增大此值，您可以使 RepAgent 执行程序线程更长时间地控制 CPU 资源，这样可以提高从 RepAgent 到 Replication Server 的吞吐量。 使用 **alter connection** 在连接级别设置此参数。 缺省值：2,147,483,647 最小值：1 最大值：2,147,483,647

配置参数	说明
exec_max_cache_size	指定要为执行程序命令高速缓存分配的的内存量。 缺省值：1,048,576 字节 对于 32 位 Replication Server： • 最小值 – 0 • 最大值 – 2,147,483,647 对于 64 位 Replication Server： • 最小值 – 0 • 最大值 – 2,251,799,813,685,247
exec_nrm_request_limit	指定可用于主数据库中等待规范化的消息的内存量。 在使用 **exec_nrm_request_limit** 之前，使用 **configure replication server** 将 **nrm_thread** 设置为“on”。 最小值：16,384 字节 最大值：2,147,483,647 字节 缺省值： • 32 位 – 1,048,576 字节 (1MB) • 64 位 – 8,388,608 字节 (8MB) 在更改 **exec_nrm_request_limit** 的配置后，挂起并恢复 Replication Agent。 许可证：在高级服务选件下单独许可。
exec_prs_num_threads	通过为来自主数据库的特定连接启动多个分析程序线程来启用异步分析程序功能，并指定连接的异步分析程序线程数。 缺省值：0 最小值：0 可禁用异步分析程序。 最大值：20 **注意：** 在配置异步分析程序之前，请确保 **smp_enable** 设置为 on，并且 Replication Server 主机可以支持使用额外的线程进行分析。
exec_sqm_write_request_limit	指定可用于等待写入到入站队列中的消息的内存量。 缺省值：1MB，最小值：16KB，最大值：2GB

配置参数	说明
md_sqm_write_request_limit	指定分配器可用于存放等待写入出站队列的消息的内存量。 **注意：** 在 Replication Server 12.1 中，**md_sqm_write_request_limit** 将替换 **md_source_memory_pool**。将保留 **md_source_memory_pool** 以便与旧版本的 Replication Server 保持兼容。 缺省值：1MB 最小值：16K 最大值：2GB
parallel_dsi	用于将并行 DSI 配置为缺省值的快捷方法。值“on”会将 **dsi_num_threads** 设置为 5、将 **dsi_num_large_xact_threads** 设置为 2、将 **dsi_serialization_method** 设置为 **wait_for_commit** 并将 **dsi_sqt_max_cache_size** 设置为 1 MB。值“off”会将并行 DSI 值设置为其缺省值。您可以将此参数设置为“on”，然后分别设置各个并行 DSI 配置参数，以对配置进行微调。 缺省值：off
sqm_async_seg_delete	将 **sqm_async_seg_delete** 设置为 on 可启用用于删除段的专用守护程序。 缺省值：on
sqm_cmd_cache_size	Replication Server 可以在 SQM 命令高速缓存中存储的已分析数据的最大大小（以字节为单位）。 32 位 Replication Server： • 缺省值 – 1,048,576 • 最小值 – 0，禁用 SQM 命令高速缓存 • 最大值 – 2,147,483,647 64 位 Replication Server： • 缺省值 – 20,971,520 • 最小值 – 0 • 最大值 – 2,251,799,813,685,247 如果 **cmd_direct_replicate** 或 **sqm_cache_enable** 为 off，Replication Server 将忽略为 **sqm_cmd_cache_size** 设置的任何值。

配置参数	说明
sqm_max_cmd_in_block	指定每个 SQM 块中已分析数据可以与之关联的最大条目数。 缺省值：320 最小值：0 最大值：4096 将 **sqm_max_cmd_in_block** 的值设置为 SQM 块中的条目数。根据数据配置文件，由于块大小是固定的，而消息大小是无法预料的，因此每个块有不同的条目数。如果设置的值过大，则会浪费内存。如果设置的值过小，则会影响复制性能。 如果 **cmd_direct_replicate** 或 **sqm_cache_enable** 为 off，Replication Server 将忽略为 **sqm_max_cmd_in_block** 设置的任何值。
use_batch_markers	如果将 **use_batch_markers** 设置为 **on**，则将执行函数字符串 **rs_batch_start** 和 **rs_batch_end**。 **注意：** 只需为复制数据服务器将此参数设置为 **on**，该服务器要求在 **rs_begin** 和 **rs_commit** 函数字符串中未包含的一批命令开始和结束时发送附加的 SQL 转换。 缺省值：off

另请参见

- 高级服务选件 （第 196 页）
- 并行 DSI 线程 （第 148 页）
- 分区规则：减少争用和增大并行度 （第 159 页）
- 指定组中的事务数 （第 145 页）
- 控制 RepAgent 执行程序可以处理的命令数 （第 143 页）

影响性能的路由参数

Replication Server 提供了一些影响性能的路由配置参数。

有关路由参数的完整列表，请参见《Replication Server 管理指南第一卷》中的“管理路由”。

表 17. 影响性能的路由参数

配置参数	说明
rsi_batch_size	请求截断点之前发送到另一 Replication Server 的字节数。 缺省值：256K 最小值：1K 最大值：128MB

配置参数	说明
rsi_packet_size	用于与其它 Replication Server 通信的包的大小（以字节为单位）。范围为 1024 到 16384。 缺省值：4096 字节
rsi_sync_interval	RSI 同步查询消息之间的秒数。Replication Server 使用这些消息使 RSI 出站队列与目标 Replication Server 同步。此值必须大于 0。缺省值：60 秒

使用调优参数的建议

提供一些提高 Replication Server 性能的基本建议。更改这些配置值是否会提高系统性能，取决于您的系统配置以及 Replication Server 在您节点上的使用方式。

设置 SQM Writer 等待的时间

可以使用 **init_sqm_write_delay** 和 **init_sqm_write_max_delay** Replication Server 配置参数设置 SQM writer 等待的时间。

在低卷系统上，将 **init_sqm_write_delay** 和 **init_sqm_write_max_delay** 设置为较小值，以便 SQM Writer 不需要等待很长时间即可写入部分充满的块。在高卷系统上，将这些参数设置为较大的值，因为 SQM Writer 很少等待填充块。

通过检查计数器 6038 – WritesTimerPop，来监控 SQM Writer 等待的频率。

通过检查以下计数器，可确定已经写入的充满块数或部分充满的块数：

- 6002 – BlocksWriten
- 6041 - BlocksFullWrite

如果与计数器 62002 – BlocksRead 相比，计数器 62006 – SleepsWriteQ 的值较大，则 SQM Reader 也必须频繁地等待下一消息块向下游传递，这样就会导致延迟。减小 **init_sqm_write_delay** 和 **init_sqm_write_max_delay** 的值，可避免 SQM Writer 在写入部分充满块之前等待很长时间。

理想情况下，计数器 62004 – BlocksReadCached 与计数器 62002 – BlocksRead 之比应该较高，而计数器 62006 – SleepsWriteQ 的值应该较小。这样的数值将表示 SQM Writer 的工作速度与 SQM Reader 的工作速度大致相等，不用从磁盘读取即可前后传递块。但是，这些参数是 Replication Server 范围的参数，调整它们可提高一个队列的效率，但同时可能会降低另一个队列效率。

高速缓存系统表

可以使用 **sts_cache_size** 和 **sts_full_cache_***table_name* Replication Server 配置参数高速缓存系统表。

您可以完全高速缓存某些系统表，以便那些表上的简单 **select** 语句不需要访问 RSSD。缺省情况下，`rs_repobjs` 和 `rs_users` 是完全高速缓存的。根据所用的复制定义

数和预订数，完全高速缓存这些表可能会大大降低 RSSD 访问要求。但是，如果 `rs_objects` 中唯一行的数量约等于 **sts_cachesize** 的值，则这些表可能已经被完全高速缓存了。

如果复制系统中有很多复制定义并且有很多复制定义更改请求，请确保 `rs_objects`、`rs_columns` 和 `rs_objfunctions` 的 **sts_full_cache** 设置为 off，因为对表的任何更改（完全高速缓存）将导致刷新该表的整个高速缓存。

.

可能被高速缓存的系统表
只能高速缓存某些系统表。

表 18. 可能被高速缓存的系统表

表			
`rs_classes`	`rs_dbsubsets`	`rs_version`	`rs_datatype`
`rs_databases`	`rs_columns`	`rs_config`	`rs_routes`
`rs_objects`	`rs_diskaffinity`	`rs_asyncfuncs`	`rs_users`
`rs_sites`	`rs_queues`	`rs_repdbs`	`rs_dbreps`
`rs_repobjs`	`rs_systext`	`rs_publications`	`rs_objfunctions`
`rs_clsfunctions`	`rs_translation`	`rs_targetobjs`	

复制定义更改过程
如果对 RSSD 进行很多更改（如创建、更改或删除复制定义或自定义函数字符串），Sybase 建议您在启动复制定义更改过程之前为 `rs_objects`、`rs_columns` 和 `rs_objfunctions` 禁用 **sts_full_cache**，然后在复制定义更改过程完成后将这些表的 **sts_full_cache** 设置为原始值。

提示： 如果有许多 RSSD 更改，请定期在 RSSD 表上执行 Adaptive Server **update statistics** 命令。对于复制定义更改请求（例如创建、更改或删除复制定义），受影响的表为 `rs_objects`、`rs_columns` 和 `rs_objfunctions`。对于函数字符串更改请求（例如创建、更改或删除函数字符串），受影响的表为 `rs_funcstrings` 和 `rs_systext`。

禁用 **sts_full_cache**，其中 *system_table_name* 是表的名称：

```
configure replication server
set sts_full_cache_system_table_name to ‘off
```

请参见《Replication Server 管理指南第一卷》的“管理复制表”的“修改复制定义”的“更改复制定义”中的“复制定义更改请求过程”。

执行程序命令高速缓存

当 Sybase RepAgent 最初发送主 Adaptive Server 数据库表的 **insert**、**delete** 或 **update** LTL 命令时，可以使用执行程序命令高速缓存来高速缓存该表的列名和数据类型。

列名和数据类型等元数据是 RepAgent 发送的表模式以及与 **insert**、**delete** 或 **update** 命令关联的数据的一部分。但是，通过使用高速缓存：

- 只有在启动 RepAgent 或重新启动到 Replication Server 的连接后首次处理该特定表的操作时，RepAgent 才会发送元数据以及与 **insert**、**update** 或 **delete** 命令关联的数据。当 Replication Agent 随后处理该表的事务时，Replication Agent 不发送表元数据。
- 如果 RepAgent 中没有足够的内存保存所有模式定义，RepAgent 可能会重新发送元数据和数据。
- 当 RepAgent 在特定表的表模式发生更改后处理该表上的修改时，RepAgent 发送表的元数据和数据，例如，在执行 Adaptive Server **alter table** 操作之后。

要复制对同一个表的后续操作，RepAgent 仅发送列数据，因为 Replication Server 执行程序命令高速缓存存储了元数据。由于减少了 RepAgent 元数据并使用 Replication Server 执行程序命令高速缓存进行高速缓存，因而提高了复制性能，这是因为高速缓存：

- 减少 RepAgent 将元数据打包到日志传送语言 (LTL) 包中所花的时间。
- 增大每个包中发送的数据量以减少网络通信量。
- 允许 RepAgent 将省下来的时间专用于扫描主数据库日志，而不是打包元数据。
- 允许 Replication Server 执行程序更高效地处理具有大量列的表。

注意： 高速缓存仅包含由 **insert**、**update** 或 **delete** 操作修改的表中的元数据。

系统要求

要减少表元数据，需要使用 LTL 740 或更高版本以及 Adaptive Server 15.7 或更高版本。

启用表元数据减少

为 Sybase RepAgent 启用表元数据减少。如果在 RepAgent 中启用了表元数据减少，Replication Server 将自动启用执行程序命令高速缓存。

1. 在 Adaptive Server 上，执行：

   ```
   sp_config_rep_agent database_name, 'ltl metadata reduction',
   'true'
   ```

   其中 *database_name* 是主 Adaptive Server 数据库。

   注意： 缺省情况下，**ltl metadata reduction** 设置为 false，并且 RepAgent for Adaptive Server 不启用表元数据减少。

2. 重新启动 RepAgent 以使更改生效：

```
sp_start_rep_agent database_name
```

设置执行程序命令高速缓存大小

可以使用 **exec_max_cache_size** 指定要为执行程序命令高速缓存分配的内存量。

如果为高速缓存分配的内存量不足，则会影响复制性能并频繁显示以下错误消息：

```
Executor Command Cache exceeds its maximum limit defined by
exec_max_cache_size (current value is current_exec_max_cache_size).
```

为避免复制性能进一步下降，请使用 **exec_max_cache_size** 在高速缓存中添加更多内存，或者在 Replication Agent 中禁用表元数据减少。

您可以为 32 位 Replication Server 设置 0 到 2,147,483,647 字节之间的值，以及为 64 位 Replication Server 设置 0 到 2,251,799,813,685,247 字节之间的值。对于 32 位和 64 位 Replication Server，缺省值为 1,048,576 字节。

例如，要在以下级别将执行程序命令高速缓存大小设置为 2,097,152 字节：

- 服务器级 – 用于到 Replication Server 的所有主数据库连接，请输入：

```
configure replication server
set exec_max_cache_size to '2097152'
```

- 连接级 – 用于特定主数据库连接，请输入：

```
alter connection to dataserver_name.database_name
set exec_max_cache_size to '2097152'
```

如果在两个级别都具有这些设置，Replication Server 始终使用连接级设置。无需重新启动 Replication Server 即可使更改生效。

稳定队列高速缓存

Replication Server 使用简单高速缓存机制来优化 I/O。此机制可降低写入延迟时间并提高读取速度，因为通常可快速从高速缓存中读取数据。

高速缓存由多个页构成，而每页都由多个相邻块构成。启动时会为每个队列分配一个高速缓存。更改页大小会更改稳定队列设备中 I/O 的大小。当某页已满时，整个页会采用一次写入操作完成写入。

在稳定队列高速缓存中，页指针会向前移动并在高速缓存末尾处返回。如果写入器已填满消息队列并在等待消息时阻塞，则 SQM 会刷新当前页。仅在刷新未填满的页时，数据块才会写入磁盘。

配置稳定队列高速缓存参数

您可以配置一些稳定队列高速缓存参数。

使用以下命令来设置服务器范围的高速缓存缺省值：

```
configure replication server set sqm_cache_enable to
"on|off"
```

使用以下命令来为队列启用或禁用高速缓存并替换服务器级设置：

```
alter queue q_number, q_type, set sqm_cache_enable to
"on|off"
```

当 **sqm_cache_enable** 参数被禁用时，SQM 模块会返回到早期机制，它维护固定的 16K；一块缓冲区。

使用以下命令来设置服务器范围的页大小缺省值：

```
configure replication server set sqm_page_size to
"num_of_blocks"
```

使用以下命令来为指定队列设置页大小：

```
alter queue q_number, q_type, set sqm_page_size to
"num_of_blocks"
```

num_of_blocks 指定页中 16K 块的数量。配置页大小也会设置 Replication Server 的 I/O 大小。例如，如果将页大小设置为 4，这指示 Replication Server 以 64K 的块写入稳定队列。

使用以下命令来设置服务器范围的高速缓存大小缺省值：

```
configure replication server set sqm_cache_size to
"num_pages"
```

使用以下命令来为指定队列设置高速缓存大小：

```
alter queue q_number, q_type, set sqm_cache_size to
"num_pages"
```

num_pages 指定高速缓存中页的数量。

所有 SQM 配置命令都是静态的，因此必须重新启动服务器，才能使这些命令生效。

有关这些配置参数的详细信息，请参见《Replication Server 参考手册》。

异步分析程序、ASCII 打包和直接命令复制

通过使用异步分析程序、ASCII 打包以及入站和出站直接命令复制功能，在数据转换和传输期间提高了整个复制过程的性能。

Replication Server 按以下顺序处理执行程序 (EXEC) 线程、分配器 (DIST) 线程以及数据服务器接口 (DSI) 线程中的数据：

1. EXEC 线程将 LTL 命令从 Replication Agent 传输到 Replication Server。
2. EXEC 线程分析 LTL 命令并使用内部分析的格式存储这些命令，然后使用二进制格式打包分析数据。
3. EXEC 线程通过入站 SQM 线程将二进制数据写入到入站队列中。
4. DIST 模块检索二进制命令，将命令恢复为原始格式，然后确定将这些命令发送到的位置。
5. 在 DIST 模块将命令发送到 DSI 模块之前，DIST 使用内部 ASCII 格式打包这些命令，然后通过出站 SQM 线程将这些命令写入到出站稳定队列中。
6. 接下来，DSI 模块读取并分析 ASCII 命令，然后将这些命令恢复为原始格式。

异步分析程序以及入站和出站直接命令复制功能提高了序列中的特定步骤的复制性能，而 ASCII 打包降低了队列存储消耗量。不过，可以通过将所有这些功能结合使用，最大限度提高性能和减少队列存储消耗量。可以将 **async_parser** 与 **alter connection** 一起使用以同时配置所有功能，而不是单独配置每个功能。

在配置异步分析程序功能之前，请确保 **smp_enable** 设置为 on，并且 Replication Server 主机可以支持使用额外的线程进行分析。

如果将 **async_parser** 设置为 on，则会：

- 启用异步分析程序 – 将 **exec_prs_num_threads** 设置为 2
- 启用 ASCII 打包 – 将 **ascii_pack_ibq** 设置为 on
- 启用入站直接命令复制 – 将 **cmd_direct_replicate** 设置为 on
- 启用出站直接命令复制 – 将 **dist_cmd_direct_replicate** 设置为 on

可以将 **async_parser** 设置为 on，然后单独设置各个参数以进行调优和平衡性能与资源消耗量。**async_parser** 的缺省值为 off。将 **async_parser** 设置为 off 可将各个参数重置为缺省设置。必须将 Replication Server 节点版本设置为 1571 或更高版本，然后才能将 **ascii_pack_ibq** 设置为 on。如果节点版本早于 1571，将 **async_parser** 设置为 on 只会设置 **exec_prs_num_threads**、**cmd_direct_replicate** 和 **dist_cmd_direct_replicate**。

在使用 **async_parser** 后，请使用 **suspend distributor** 和 **resume distributor** 重新启动分配器。在将 **async_parser** 设置为 on 时，Replication Server 将重新启动关联的 Replication Agent。

异步分析程序

配置额外的执行程序线程以分析来自 Replication Agent 的命令，以减少 Replication Agent 等待执行程序的时间。

在使用单个关联的执行程序线程时，Replication Agent 必须等待该线程完成分析 LTL 命令，然后 Replication Agent 才能将下一批命令传输到 Replication Server。如果配置多个专用于分析 LTL 命令的线程，Replication Server 将为专用异步分析程序线程分配单独的分析任务，从而导致并行分析几个 LTL 命令包。此外，还会提高复制吞吐量，因为 Replication Agent 缩短了它必须等待执行程序线程的时间，因为数据传输和分析是使用多个线程异步完成的。

在配置异步分析程序功能之前，请确保 **smp_enable** 设置为 on，并且 Replication Server 主机可以支持使用额外的线程进行分析。要使用异步分析程序功能，请使用 **alter connection** 设置 **exec_prs_num_threads** 以便为来自主数据库的特定连接启动多个分析程序线程，以及为该连接指定异步分析程序线程数。在设置 **exec_prs_num_threads** 时，Replication Server 将重新启动 Replication Agent。可以启动的最大线程数是 20 个。设置为 0 可禁用异步分析程序。最小值为 0（缺省值）。除了异步分析程序线程以外，Replication Server 还启动命令批处理同步线程。

在下列情况下，设置 **exec_prs_num_threads** 失败并且 Replication Server 关闭：

- 要启动的线程总数大于池中使用 **num_threads** 指定的可用线程数

- 要创建的消息队列总数超过池中使用 **num_msg_queues** 指定的可用消息队列数

示例

要为到 TOKYO_DS 数据服务器上的 pdb1 主数据库的连接启动四个分析程序线程，请输入：

```
alter connection to TOKYO_DS.pdb1
set exec_prs_num_threads to 4
go
```

检查异步分析程序线程
如果 **exec_prs_num_threads** 设置为大于 0 的值，可以使用 **admin who** 显示存在的异步分析程序线程数。此外，**admin who** 还会显示存在一个命令批处理同步线程。

ASCII 打包
通过将 ASCII 打包与异步分析程序一起使用，减少入站队列中的打包命令所消耗的稳定队列存储空间。

请将 **ascii_pack_ibq** 设置为 on，以使 Replication Server 使用 ASCII 打包对入站队列中的命令进行打包，而不是对这些命令使用缺省二进制打包模式，前者消耗的稳定队列存储空间比二进制打包少。缺省值为 off。虽然二进制打包的命令消耗更多的稳定队列存储空间，但 Replication Server 解释二进制打包命令的速度比 ASCII 打包命令快。

如果将 **ascii_pack_ibq** 设置为 on，则仅对入站队列的命令进行打包。无法更改出站队列中的命令的打包模式，因为 Replication Server 已使用 ASCII 包对这些命令进行打包。必须为 Replication Server 启用异步分析程序功能，以便从入站队列的 ASCII 打包中受益，并且必须在将 **ascii_pack_ibq** 设置为 on 之前将 Replication Server 节点版本设置为 1571 或更高版本。

示例

要将来自 TOKYO_DS 数据服务器上的 pdb1 主数据库的连接的打包模式设置为 ASCII 打包，请输入：

```
alter connection to TOKYO_DS.pdb1
set ascii_pack_ibq to on
go
```

入站命令的直接复制
减少 Replication Server EXEC 和 DIST 模块之间的入站复制路径中的命令转换以提高复制性能。

对于入站数据，您可以将 **cmd_direct_replicate** 设置为 on，以使执行程序线程将内部分析数据与二进制或 ASCII 格式数据一起发送。Replication Server 将分析数据存储到单独的 SQM 命令高速缓存中。SQM 命令高速缓存中的分析数据映射到 SQM 高速缓存中存储的二进制或 ASCII 格式数据。如果需要，分配器模块可以直接检索并处理内部格式分析数据，从而省去了分析二进制或 ASCII 格式数据所花的时间。由于分析 ASCII 格式会消耗更多的资源，因此，ASCII 格式打包数据减少的命令转换和后续提高的复制性能比二进制打包数据多。缺省值为 off。

只有在分配器线程最初可以从 SQM 高速缓存中读取而不是从物理磁盘中读取，并且使用 **sqm_cache_size** 设置了相应的 SQM 高速缓存大小时，该线程才能从 SQM 命令高速缓存中读取。在分配器线程从 SQM 高速缓存中读取命令后，该线程能否在 SQM 命令高速缓存中找到该命令的已分析版本取决于您使用 **sqm_cmd_cache_size** 设置的 SQM 命令高速缓存大小以及 SQM 块中可以与分析的命令关联的最大条目数（可使用 **sqm_max_cmd_in_block** 进行设置）。

可以使用 **configure replication server** 在服务器级别为到数据库的所有 Replication Agent 连接设置 **cmd_direct_replicate**。否则，请使用 **alter connection** 为各个连接设置该参数。如果使用 **alter connection**，请重新启动 Replication Agent 以使配置生效。如果使用 **configure replication server**，请重新启动 Replication Server。

可以使用 **sqm_cmd_cache_size** 和 **sqm_max_cmd_in_block** 参数设置 SQM 命令高速缓存配置。您可以在同一命令中配置 **cmd_direct_replicate**、**sqm_cmd_cache_size** 和 **sqm_max_cmd_in_block**，也可以单独进行配置。如果 **sqm_cache_enable** 为 off，Replication Server 将忽略 **sqm_cmd_cache_size** 和 **sqm_max_cmd_in_block** 设置。

示例 1

要为 64 位 Replication Server 的所有连接和队列设置配置，请输入：

```
configure replication server
set cmd_direct_replicate to 'on'
set sqm_cmd_cache_size to '40971520'
set sqm_max_cmd_in_block to '640'
go
```

示例 2

要为到 TOKYO_DS 数据服务器上的 `pdb1` 主数据库的连接以及 32 位 Replication Server 的入站队列编号 2 设置配置，请输入：

```
alter connection to TOKYO_DS.pdb1
set cmd_direct_replicate to 'on'
go
alter queue 2, 1,
set sqm_cmd_cache_size to '2048576'
set sqm_max_cmd_in_block to '640'
go
```

另请参见

- 增加队列块大小 （第 212 页）
- 使用计数器监控性能 （第 251 页）
- 配置稳定队列高速缓存参数 （第 134 页）

使用 SQM 命令高速缓存计数器监控性能

如果 **sqm_cache_enable** 和 **cmd_direct_replicate** 为 on，并且 **sqm_cmd_cache_size** 和 **sqm_max_cmd_in_block** 设置为非零值，则可以使用几个计数器监控复制性能，因为执行程序和分配器线程与分析数据进行交互。

表 19. SQM 命令高速缓存计数器

计数器	说明
RACmdsDirectRep-Send	从执行程序线程中发送的与分析数据关联的命令数。
DISTCmdsDirectRe-pRecv	分配器直接从执行程序中收到的具有与语句关联的分析数据的命令数，可能会在其中跳过分析处理。
SQMNoDirectRepli-cateInCache	从执行程序线程中发送的具有分析数据的命令数，但无法沿复制路径向分配器进一步发送分析的数据，因为命令高速缓存超过 **sqm_cmd_cache_size**。
SQMNoDirectRepli-cateInSQMCache	从执行程序线程中发送的具有分析数据的命令数，但无法沿复制路径向分配器进一步发送分析的数据，因为在读取之前在 SQM 高速缓存中覆盖了这些命令。
SQMNoDirectRepli-cateInBlock	从执行程序线程中发送的具有分析数据的命令数，但无法沿复制路径向分配器进一步发送分析的数据，因为当前 SQM 块的分析数据条目数超过 **sqm_max_cmd_in_block**。

出站命令的直接复制

减少 Replication Server DIST 和 DSI 模块之间的出站复制路径中的命令转换以提高复制性能。

对于出站数据，您可以将 **dist_cmd_direct_replicate** 设置为 on，以允许 DIST 模块将内部分析数据与打包的 ASCII 格式数据一起发送。如果需要，DSI 模块可以直接从分析数据中检索数据并进行处理，从而省去了分析 ASCII 格式数据所花的时间。如果将 **dist_cmd_direct_replicate** 设置为 off，DIST 模块仅将打包的 ASCII 数据发送到 DSI。缺省值是 on。

只有在 DSI 模块线程最初可以从 SQM 高速缓存中读取而不是从物理磁盘中读取，并且使用 **sqm_cache_size** 设置了相应的 SQM 高速缓存大小时，该模块才能从 SQM 命令高速缓存中读取。在 DSI 模块从 SQM 高速缓存中读取命令后，该模块能否在 SQM 命令高速缓存中找到该命令的已分析版本取决于您使用 **sqm_cmd_cache_size** 设置的 SQM 命令高速缓存大小以及 SQM 块中可以与分析的命令关联的最大条目数（可使用 **sqm_max_cmd_in_block** 进行设置）。

可以使用 **configure replication server** 在服务器级别为所有分配器设置 **dist_cmd_direct_replicate**。否则，请使用 **alter connection** 为各个分配器设置该参数。在使用 **alter connecion** 时，请使用 **suspend distributor** 和 **resume distributor** 重新启动

特定的分配器以使配置生效。如果使用 **configure replication server**，请重新启动 Replication Server。

可以使用 **sqm_cmd_cache_size** 和 **sqm_max_cmd_in_block** 参数设置 SQM 命令高速缓存配置。您可以在同一命令中配置 **dist_cmd_direct_replicate**、**sqm_cmd_cache_size** 和 **sqm_max_cmd_in_block**，也可以单独进行配置。如果 **sqm_cache_enable** 为 off，Replication Server 将忽略 **sqm_cmd_cache_size** 和 **sqm_max_cmd_in_block** 设置。

示例 1

要为 64 位 Replication Server 的所有连接和队列设置配置，请输入：

```
configure replication server
set dist_cmd_direct_replicate to 'on'
set sqm_cmd_cache_size to '40971520'
set sqm_max_cmd_in_block to '640'
go
```

示例 2

要为到 TOKYO_DS 数据服务器上的 pdb1 主数据库的连接以及 32 位 Replication Server 的入站队列编号 3 设置配置，请输入：

```
alter connection to TOKYO_DS.pdb1
set dist_cmd_direct_replicate to 'on'
go
alter queue 2, 1,
set sqm_cmd_cache_size to '2048576'
set sqm_max_cmd_in_block to '640'
go
```

使用 SQM 命令高速缓存计数器监控性能

如果 **sqm_cache_enable** 和 **dist_cmd_direct_replicate** 为 on，并且 **sqm_cmd_cache_size** 和 **sqm_max_cmd_in_block** 设置为非零值，则可以使用几个计数器监控复制性能，因为 EXEC、DIST 和 DSI 线程和模块与分析数据进行交互。

表 20. SQM 命令高速缓存计数器

计数器	说明
DISTCmdsDirectRepSend	直接从 DIST 模块发送的采用分析数据格式的命令数。
DSIECmdsDirectRepRecv	DSI-E 线程直接从 DIST 模块（传送出站命令）或 EXEC 模块（传送备用入站命令）接收的命令数。如果将 DSI 模块用于热备份连接，则 DSIECmdsDirectRepRecv 计入入站命令，否则，DSIECmdsDirectRepRecv 计入出站命令。
SQMNoDirectReplicateInCache	从分配器线程中发送的具有分析数据的命令数，但无法沿复制路径向 DSI 线程进一步发送分析的数据，因为命令高速缓存超过 **sqm_cmd_cache_size**。

计数器	说明
SQMNoDirectReplicateInSQMCache	从分配器线程中发送的具有分析数据的命令数，但无法沿复制路径向 DSI 线程进一步发送分析的数据，因为在读取之前在 SQM 高速缓存中覆盖了这些命令。
SQMNoDirectReplicateInBlock	从分配器线程中发送的具有分析数据的命令数，但无法沿复制路径向 DSI 线程进一步发送分析的数据，因为当前 SQM 块的分析数据条目数超过 **sqm_max_cmd_in_block**。

SQM 命令高速缓存配置

SQM 命令高速缓存配置设置取决于可用于 Replication Server 的总内存量、入站和出站队列数以及事务配置文件（取决于命令大小）。

在设置 SQM 命令高速缓存配置时：

- 如果 Replication Server 具有较大的总 SQM 高速缓存，请增加 **sqm_cmd_cache_size**。总 SQM 高速缓存 = **sqm_cache_size**（页）* **sqm_page_size**（块）* **block_size** (KB)
- 如果命令大小或表行大小较大，请减少 **sqm_max_cmd_in_block**。
- 如果 **block_size** 较大，请增加 **sqm_max_cmd_in_block**。

在设置初始值后，请根据监控计数器中的复制性能和数据调整这些值：

- 如果 SQMNoDirectReplicateInCache 显示较大的值，请增加 **sqm_cmd_cache_size**。
- 如果 SQMNoDirectReplicateInBlock 显示较大的值，请增加 **sqm_max_cmd_in_block**。

可以使用 **configure replication server** 为到 Replication Server 的所有数据库连接更改 **sqm_cache_size**、**sqm_page_size** 和 **block_size**。否则，使用 **alter queue** 为特定数据库连接设置配置。

有关参数缺省值和有效的值范围，请参见《Replication Server 参考手册》中的“Replication Server 命令”。

设置唤醒时间间隔

可以使用 **rec_daemon_sleep_time** 和 **sub_daemon_sleep_time** Replication Server 配置参数设置唤醒时间间隔。

缺省情况下，恢复守护程序和预订守护程序每隔两分钟唤醒一次，以检查 RSSD 中是否存在消息。在通常的生产环境中，预订守护程序很少使用。因此，您也许能够将预订守护程序的唤醒时间间隔设置为最大值：31,536,000 秒。同样，您可以评估是否要将恢复守护程序的唤醒时间间隔设置为较长时间。

设置 SQT 高速缓存大小

可以使用 **sqt_max_cache_size** Replication Server 配置参数和 **dsi_sqt_max_cache_size** 数据库连接配置参数设置 SQT 高速缓存大小。

通过检查计数器 24005 – CacheMemUsed 监控 SQT 高速缓存使用情况。相反，应监控计数器 24009 – TransRemoved。如果 TransRemoved 保持为零，则表明没有从高速缓存刷新事务来为其它事务腾出空间，这时您不需要调整 **sqt_max_cache_size**。

警告！ 如果服务器 **memory_limit** 未设置足够大以容纳 sqt 高速缓存大小，则将 **sqt_max_cache_size** 设置太高可能导致服务器关闭并可能影响 Replication Server 的整个资源。

sqt_max_cache_size 适用于支持 DIST 客户端的所有 SQT 高速缓存，并为支持 DSI 客户端的 SQT 高速缓存提供缺省值。DIST 可以快速完成事务；其 SQT 高速缓存不需要与 DSI 的 SQT 高速缓存一样大。因此，建议使用连接配置参数 **dsi_sqt_max_cache_size** 逐个设置 DSI 的 SQT 高速缓存大小，并仅对 DIST SQT 高速缓存使用 **sqt_max_cache_size**。

注意： 在早于 15.5 的 Replication Server 版本中，将 **sqt_max_cache_size** 设置过高可能会降低复制速度。此建议不适用于 Replication Server 15.5 和更高版本。

控制未完成的字节数

可以使用 **exec_nrm_request_limit**、**exec_sqm_write_request_limit** 和 **md_sqm_write_request_limit** 数据库连接配置参数控制内存的未完成字节数。

exec_nrm_request_limit 是单独许可的选项，可用于提高 RepAgent 执行程序线程的效率。

另请参见

- 提高的 RepAgent 执行程序线程效率 （第 211 页）

exec_sqm_write_request_limit 数据库配置参数

exec_sqm_write_request_limit 控制可用于等待写入到入站队列中的消息的内存量。

md_sqm_write_request_limit 数据库配置参数

md_sqm_write_request_limit 控制 DIST 线程必须等待其中一些字节写入出站队列之前，可以包含的未完成字节数。

使用计数器监控性能

可以使用计数器监控 RepAgent 执行程序和 NRM 线程性能。

通过检查此计数器，可以监控 RepAgent 执行程序在等待规范化完成的过程中休眠的次数和持续时间：

- 58038 – RAWaitNRMTime

通过检查此计数器，可以监控线程（可能是 RepAgent 执行程序或 NRM）等待消息进入入站队列之前休眠的次数和持续时间：

- 58019 – RAWriteWaitsTime

如果 RAWriteWaitsTime 始终较大，请检查 StableDevice I/O。

另请参见

- 使用计数器监控性能 （第 251 页）

控制网络操作数

可以使用 **dsi_cmd_batch_size** 数据库连接配置参数控制 DSI 命令批处理的大小。

dsi_cmd_batch_size 控制 DSI 用来将命令发送到复制数据服务器的缓冲区的大小。当 DSI 配置批处理设置为“on”时，DSI 在将它发送到复制数据服务器之前，会尽可能多地放置适合一个命令批处理的命令。在某些情况下，增大 **dsi_cmd_batch_size** 的值，可让复制数据库的每个命令批处理包含更多的工作，从而提高吞吐量。

用于监控批处理和批处理大小的计数器

Replication Server 提供了一些计数器以监控批处理和批处理大小。

通过查看计数器 57076 – DSIEBatchSize，您可以监控某个批处理的平均大小。通过查看计数器 57070 – DSIEBatchTime，您可以监控处理批处理所用的平均时间（从创建批处理开始到刷新批处理并处理完结果的时间）。

在监控批处理和批处理大小的效果方面，也可以使用以下计数器：

57037 – SendTime	57079 – DSIEOCmdCount	57063 – DSIEResultTime
57070 – DSIEBatchTime	57092- DSIEBFMaxBytes	57076 – DSIEBatchSize

控制 RepAgent 执行程序可以处理的命令数

可以使用 **exec_cmds_per_timeslice** 数据库连接配置参数控制 RepAgent 执行程序线程可以处理的命令数。

缺省情况下，**exec_cmds_per_timeslice** 参数的值是 2,147,483,647，表明 RepAgent 执行程序线程在必须将 CPU 让予其它线程之前能够处理的命令不得超过 2,147,483,647 个。根据您的环境，增大或减小这些值可能会提高性能。

如果入站队列的处理速度慢，请尝试增大这些值，以便为 RepAgent 执行程序线程和 DIST 线程提供更多的时间来执行其工作。但是，如果出站队列的处理速度慢，请尝试减小这些参数值，以便 DSI 有更多的工作时间。

如果 CPU 资源受 Replication Server 支持的连接数的限制，则增大 **exec_cmds_per_timeslice** 的值可能会导致总体性能下降。在这种情况下，如果为 RepAgent 执行程序提供对 CPU 资源的更多控制，可能会减少提供给其它 Replication Server 线程的资源。

使用此计数器可监控 RepAgent 执行程序线程让出 CPU 的次数和持续时间：

- 58016 – RAYieldTime

指定分配的稳定队列段数

可以使用 **sqm_recover_segs** Replication Server 配置参数指定 Replication Server 在使用恢复 QID 信息更新 RSSD 之前分配的稳定队列段数。

如果将 **sqm_recover_segs** 设置为较小的值，则执行更多的 RSSD 更新，这可能会降低性能。如果将 **sqm_recover_segs** 设置为较大的值，则执行较少的 RSSD 更新，这可能会提高性能，但恢复时间较长。

通过检查计数器 6036 – UpdsRsoqid，监控 SQM Writer 对 `rs_oqids` 表进行更新的频率。通常，增大 **sqm_recover_segs** 的值，可减少分配段所需的时间和系统资源，从而提高性能。但是，队列启动和重新启动所用时间较长，因为 SQM Writer 必须扫描队列的更多部分才能确定为每个源成功写入了最后一条消息。每个段都需要 1MB 的队列空间；通过计算在启动或重新启动时 SQM Writer 能够用于扫描的兆字节数来确定 **sqm_recover_segs** 的值。例如，如果 SQM Writer 可以扫描 50MB 的队列，且不会降低 Replication Server 的启动或重新启动速度，则将 **sqm_recover_segs** 设置为 50。

为稳定队列选择磁盘分区

可以使用 **disk_affinity** 数据库连接配置参数指定在当前分区已满时应将下一个段分配到的分区的逻辑名称。

通过 Replication Server 分区关联功能可选择 Replication Server 要为其分配稳定队列段的磁盘分区。Sybase 建议，为了提高总体吞吐量，应将速度较快的设备与处理速度较慢的稳定队列关联。

另请参见

- 分配队列段 （第 248 页）

使 SMP 更有效

可以使用 **smp_enable** Replication Server 配置参数启用对称多重处理 (SMP)。

若要确定有效使用 SMP 所需的处理器数，请以两个处理器为基础，另外为每四个队列添加一个处理器。处理器速度可确定这些数量是否正确（即满足您的性能需要）。如果有支持并行 DSI 的出站队列，而且有超过 12 个的 DSI 执行程序线程，则您可能需要增大出站队列的处理器/线程比 — 每 3 个（或者甚至 2 个）出站队列对应 1 个处理器。

Replication Server 总是根据支持的连接数和路由数来使用有限数目的线程。即使所有的线程都始终处于繁忙状态，使越来越多的处理器可供 Replication Server 使用最终也将导致“CPU 饱和”，超过此饱和度，另外添加的处理器将不会提高性能。此时，解决您所遇到的由 CPU 资源而导致的任何性能问题的最好方法是引入运行速度更快的 CPU。

在某些情况下，有证据表明为 Replication Server 提供太多的处理器实际上会降低性能。在这样的情况下，问题似乎出现在可用处理器之间强制线程环境切换所用的时间上。使用操作系统 (OS) 监控实用程序，可以监控操作系统对 Replication Server 进程及其线程的管理。这些实用程序将有助于确定减少可用于 Replication Server 的 CPU 是否会减少这样的环境切换数。

指定组中的事务数

您可以使用不同的配置参数控制组中的事务数。

数据库配置参数：dsi_max_xacts_in_group

可以使用 **dsi_max_xacts_in_group** 指定组中的最大事务数。

较大的数值可能会减少复制数据库上的提交处理，从而提高吞吐量。

使用 **dsi_max_xacts_in_group** 可以控制组大小。将 **dsi_xact_group_size** 设置为最大值 2,147,483,647，而且不要更改它。通过将 **dsi_max_cacts_in_group** 的值减至 1（指示不分组），可能会减少并行事务之间的争用。

通过检查计数器 57001 – UnGroupedTransSched 可监控每个 DSI-E 线程放置在组中的平均事务数。

通过检查以下计数器，为总 DSI 连接监控每个组中的平均事务数：

- 5000 – DSIReadTranGroups
- 5002 – DSIReadTransUngrouped

通过检查以下计数器，可监控关闭组的原因：

- 5042 – GroupsClosedBytes
- 5043 – GroupsClosedNoneOrig
- 5044 – GroupsClosedMixedUser
- 5045 – GroupsClosedMixedMode
- 5049 – GroupsClosedTranPartRule
- 5051 – UserRuleMatchGroup
- 5053 – TimeRuleMatchGroup
- 5055 – NameRuleMatchGroup
- 5063 – GroupsClosedTrans
- 5068 – GroupsClosedLarge
- 5069 – GroupsClosedWSBSpec
- 5070 – GroupsClosedResume
- 5071 – GroupsClosedSpecial
- 5072 – OriginRuleMatchGroup
- 5074 – OSessIDRuleMatchGroup
- 5076 – IgOrigRuleMarchGroup

数据库配置参数：dsi_xact_group_size 和 dsi_max_xacts_in_group

将这些配置参数一起使用，以增大可以组合为一个事务的事务数，以便应用于复制数据库。

如果每个事务的平均命令数较小（5 或更小），则可以使用 **dsi_xact_group_size** 和 **dsi_max_xact_in_group** 增加事务应用时间。

Sybase 建议将 **dsi_xact_group_size** 设置为最大值，并使用 **dsi_max_xact_in_group** 控制事务组大小。

设置事务大小

对于单个 DSI 连接，请将 **dsi_large_xact_size** 值设置为最大值 2,147,483,647。即使在未配置并行 DSI 时，DSI/S 也会读取 **dsi_large_xact_size** 设置的语句限制并执行一些与并行 DSI 有关的任务。

启用非阻塞提交

可以使用 **dsi_non_blocking_commit** Replication Server 配置参数启用非阻塞提交，即，指定在提交后将 Replication Server 保存消息的时间延长的分钟数。

如果 Adaptive Server 15.0 和更高版本中提供了延迟提交功能，或者 Oracle 10g v2 中提供了等效延迟提交功能，非阻塞提交功能可提高复制性能。

值的范围：0 – 60 分钟。

缺省值：0 – 禁用非阻塞提交。

内存消耗量控制

在内存消耗量超过定义的阈值并且您可以控制 EXEC、DSI 和 SQT 使用的内存时，Replication Server 可能会显示警告消息。

内存阈值警告消息

将 Replication Server 配置为在内存消耗量超过定义的总可用内存阈值百分比时显示警告消息。

要配置警告消息，请使用：

- **mem_warning_thr1** – 指定在第一个警告消息生成之前使用的总内存阈值百分比。
  缺省值：**memory_limit** 值的 80%。
  范围：1 – 100.
- **mem_warning_thr2** – 指定在第二个警告消息生成之前使用的总内存阈值百分比。
  缺省值：**memory_limit** 值的 90%。
  范围：1 – 100.

Replication Server 线程内存量控制

您可以在 Replication Server 线程消耗的内存量超过 **memory_limit** 定义的可用内存时避免自动关闭 Replication Server。

在 Replication Server 中，需要大量内存的线程是：

- DSI
- EXEC
- SQT

这些线程在接收或处理新数据之前通过执行内存使用量检查来执行内存控制。在内存控制期间，如果内存使用量很高，则会通过以下措施来调整线程运行情况：

- 阻止线程对新数据进行分组，并清除和处理现有数据；或者，
- 使线程进入休眠状态，以使它在内存可用之前不接收新数据。

要在 EXEC、DST 和 SQT 线程中管理流控制，请使用：

- **mem_thr_dsi** – 指定用于强制 DSI 线程停止填充 SQT 高速缓存的总内存的百分比。
  缺省值：**memory_limit** 值的 80%。
- **mem_thr_exec** – 指定用于强制 EXEC 线程停止接收来自 RepAgent 的命令的总内存的百分比。
  缺省值：**memory_limit** 值的 90%。
- **mem_thr_dsi** – 指定用于强制 SQT 线程刷新其高速缓存中的最大事务的总内存的百分比。
  缺省值：**memory_limit** 值的 85%。

可以使用 **memory_control** 管理线程的内存控制行为。**memory_control** 的有效值是 enable（缺省值）或 disable。这样，Replication Server 便可控制内存消耗量，而且不会由于内存问题而关闭。

可以使用 **configure replication server** 更改配置参数的缺省值。可以使用 **admin config** 查看缺省值或现有值。

请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**configure replication server**”。

监控线程信息

可以使用 **admin who** 提供有关线程内存控制行为的信息。

状态	说明
正在控制内存	线程正在执行内存控制。
正在为内存而休眠	线程正在休眠，直到内存可用。

请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**admin who**”。

内存管理统计信息

可以使用 **admin stats** 查看内存管理统计信息。

内存计数器在 `rsh` 模块中处于启用状态。若要报告内存计数器，请使用：

```
admin stats,rsh display_name instance_id
```

其中：

- *display_name* – 是计数器的名称。使用 **rs_helpcounter** 可获取有效的显示名。*display_name* 只与 *module_name* 一起使用。
- *instance_id* – 标识模块（如 SQT 或 SQM）的特定实例。若要查看实例 ID，请执行 **admin who** 并查看 *Info* 列。对于 `rsh` 模块，必须使用 *SPID*。若要查看 *SPID*，请执行 **admin who** 并查看 *Spid* 列。

请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**admin stats**”。

并行 DSI 线程

您可以配置数据库连接，以便使用并行 DSI 线程（而不是单个 DSI 线程），将事务应用于复制数据服务器。

并行应用事务可加快复制速度，但会保持在主节点上发生的事务的串行提交顺序。

当并行 DSI 线程处于活动状态时，Replication Server 通常在上一事务提交之前、在 DSI 获得下一事务的提交记录*之后*开始处理事务。提交将被延迟，直到确定已经提交了所有的以前事务。Replication Server 可以使用以下任一方法，维持事务的提交顺序并检测以并行方式同时执行的事务中的更新冲突：

- 在内部，使用 Replication Server 内部表和函数字符串，或者
- 在外部，使用复制数据库中的 `rs_threads` 系统表。

Replication Server 可以像使用并行 DSI 线程处理包含大量操作的事务那样实现另外的并行度。在 DSI 获得提交记录之前 开始处理大事务。尽管这意味着可以更快地处理大事务，但也意味着在热备份情况下，Replication Server 可能会开始处理最终将被回退的事务。不过，通过使用预订复制，DIST 线程将捕获回退事务。

Replication Server 提供了用于最大限度地提高并行度和将事务之间的争用减到最小的其它选项。例如：

- 使用事务序列化方法，您可以选择系统能够处理且不会导致冲突的并行度。
- 事务分区规则提供了其它调优，从而影响事务的分组和分配方式，以避免复制数据库中发生争用。

使用并行 DSI 线程的优点和风险

对于大多数主数据库，许多用户和应用程序都可以同时创建事务。通过一个连接将所有这些事务都传递到复制数据库，可能会产生严重的瓶颈。此瓶颈可能会导致主数据库和复制数据库之间出现不必要的延迟。

在 Replication Server 内启用并行 DSI 的优点是，通过同时处理跨多个复制数据库的多个事务，减少了此类潜在的瓶颈。

启用并行 DSI 的风险是，引入了多个复制连接及其事务之间的争用。针对复制同时应用事务可能会引起事务对复制资源的争用，从而产生不同类型的瓶颈。

因此，成功使用并行 DSI 线程需要深入了解复制环境和进行迭代测试，以确定哪些并行 DSI 调优参数最有益。目标是提供高吞吐量，同时控制在复制数据库中引入的争用量。

例如，有一项包括必须复制的 1000 个事务的工作。通过一个复制连接发送所有 1000 个事务将需要一段时间。但是，尝试配置和使用 1000 个连接（每个事务一个连接），将很可能导致争用并使服务器资源紧张。一个成功的配置需要在这两种情况之间取得平衡；它既取决于事务配置文件，也取决于发出这些事务对使用并行 DSI 的复制的影响。

在第二个示例中，在主数据库上发出的两个串行事务中的每一个都会对同一个表行执行一个更新操作。如果两个连接在复制数据库中并行尝试这两个事务，则将排它性访问权限授予访问表行的第一个事务。第二个事务必须等待，直到第一个事务完成提交或回退并因而释放行。虽然最终应用了这两个事务，但是并行 DSI 配置没有带来任何优势。事务将以串行方式处理，与没有并行 DSI 时处理它们一样。争用已经抵消了使用并行 DSI 的任何优点。

并行 DSI 参数

您可以自定义并行 DSI 线程环境。

将这些配置参数与 **alter connection** 一起使用，可以调优单个连接的并行 DSI 线程。

若要为并行 DSI 配置连接，请将 **parallel_dsi** 参数设置为 **on**，然后设置单个并行 DSI 配置参数，以便对环境进行微调。

例如，若要为到 SYDNEY_DS 数据服务器中的 `pubs2` 数据库的连接启用并行 DSI，请输入：

```
alter connection to SYDNEY_DS.pubs2
 set parallel_dsi to 'on'
```

注意： 还可以使用 **configure replication server** 命令设置各个并行 DSI 配置参数。

并行 DSI 配置参数

Replication Server 提供了一些并行 DSI 配置参数。

表 21. 并行 DSI 配置参数

参数	说明
dsi_commit_check_locks_intrvl	DSI 执行程序线程在 `rs_dsi_check_thread_lock` 函数字符串的执行之间等待的毫秒数 (ms)。 缺省值：1000 毫秒（1 秒） 最小值：0 最大值：86,400,000 毫秒（24 小时）
dsi_commit_check_locks_log	在记录警告消息之前，DSI 执行程序线程执行 `rs_dsi_check_thread_lock` 函数字符串的次数。 缺省值：200 最小值：1 最大值：1,000,000
dsi_commit_check_locks_max	在回退并重试事务之前，DSI 执行程序线程执行 `rs_dsi_check_thread_lock` 函数字符串的最多次数。 缺省值：400 最小值：1 最大值：1,000,000
dsi_commit_control	指定提交控制处理是由 Replication Server 使用内部表在内部进行处理 (on)，还是使用 `rs_threads` 系统表在外部进行处理 (off)。 缺省值：on
dsi_ignore_underscore_names	当 **dsi_partitioning_rule** 设置为“name”时，指定 Replication Server 是否忽略以下划线开头的事务名称。值为“on”和“off”。 缺省值：on

参数	说明
dsi_isolation_level	指定事务的隔离级别。ANSI 标准和 Adaptive Server 支持的值包括： • 0 – 确保一个事务所写入的数据表示实际数据。 • 1 – 防止脏读，并确保一个事务所写入的数据表示实际数据。 • 2 – 防止非重复读取和脏读，并确保一个事务所写入的数据表示实际数据。 • 3 – 防止幻像行、非重复读取和脏读，并确保一个事务所写入的数据表示实际数据。 通过使用自定义函数字符串，Replication Server 可以支持复制数据服务器可使用的任何隔离级别。不再仅限于支持 ANSI 标准。 缺省值：目标数据服务器的当前事务隔离级别
dsi_large_xact_size	事务中所允许的语句数，如果超过该值，事务将被视为大事务。 缺省值：100 最小值：4 最大值：2,147,483,647（以字节为单位）
dsi_max_xacts_in_group	指定组中事务的最大数量。较大的数字可以改善复制数据库的数据延迟问题。 值的范围：1 – 1000。缺省值：20
dsi_max_cmds_in_batch	定义可对其输出命令进行批处理的源命令的最大数。 范围：1 – 1000 缺省值：100
dsi_num_large_xact_threads	为用于大事务而保留的并行 DSI 线程的数量。最大值是 **dsi_num_threads** 的值减去 1。 缺省值：0
dsi_num_threads	要用于连接的并行 DSI 线程数。如果值为 1，则禁用并行 DSI 功能。 缺省值：1 最小值：1 最大值：255
dsi_partitioning_rule	指定 DSI 用来将可用的并行 DSI 线程中的事务分区的分区规则（一个或多个）。值为 **origin**、**origin_sessid**、**time**、**user**、**name**、**none** 和 **ignore_origin**。 缺省值：none

参数	说明
dsi_serialization_method	指定用于确定何时可以启动事务而又仍保持一致性的方法。在任何情况下，都会保留提交顺序。 这些选项方法按并行度的大小从高到低进行排序。并行度越大，将会导致在将并行事务应用于复制数据库时，这些并行事务之间的争用越多。若要减少争用，请使用 **dsi_partition_rule** 选项。 • **no_wait** – 指定事务一旦就绪即可启动，而不考虑其它事务的状态。 **注意：** 如果 **dsi_commit_control** 设置为“on”，则只能将 **dsi_serialization_method** 设置为 **no_wait**。 • **wait_for_start** – 指定某个事务可以在预定在它紧前面提交的事务启动后立即启动。 • **wait_for_commit**（缺省值） – 指定事务直到安排在其紧前面提交的事务准备好提交后才可开始。 • **wait_after_commit** – 指定一个事务要等到其前一个预定提交的事务已完全提交后才启动。 保留下面这些选项，仅仅是为了向后兼容早期版本的 Replication Server： • **none** – 与 **wait_for_start** 相同。 • **single_transaction_per_origin** – 与 **dsi_partitioning_rule** 设置为 **origin** 的 **wait_for_start** 相同。 • **isolation_level_3** – 与 **dsi_isolation_level** 设置为 **3** 的 **wait_for_start** 相同。
dsi_sqt_max_cache_size	出站队列的最大 SQT 高速缓存大小（以字节为单位）。缺省值为 0，表示 **sqt_max_cache_size** 参数的当前设置用作连接的高速缓存最大大小。 缺省值：0 对于 32 位 Replication Server： • 最小值 – 0 • 最大值 – 2,147,483,647（以字节为单位） 对于 64 位 Replication Server： • 最小值 – 0 • 最大值 – 2,251,799,813,685,247（以字节为单位）

参数	说明
parallel_dsi	配置并行 DSI 线程的速记方法。值“on”将 **dsi_num_threads** 设置为 5，将 **dsi_num_large_xact_threads** 设置为 2，将 **dsi_serialization_method** 设置为 **wait_for_commit** 并将 **dsi_sqt_max_cache_size** 设置为 1 MB（在 32 位平台上）或 20 MB（在 64 位平台上）。值“off”会将并行 DSI 值设置为其缺省值。您可以将此参数设置为“on”，然后分别设置各个并行 DSI 配置参数，以对配置进行微调。 缺省值：off

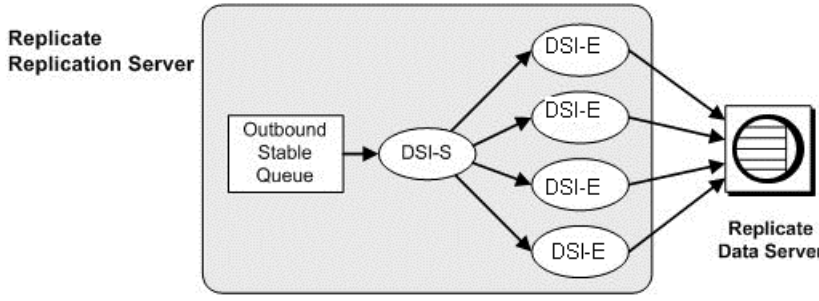
另请参见

- 分区规则：减少争用和增大并行度 （第 159 页）
- 设置 SQT 高速缓存大小 （第 142 页）
- 配置并行 DSI 以优化性能 （第 167 页）

并行 DSI 的组件

了解并行 DSI 的组件。

图 13：并行 DSI 的组件



DSI 调度程序线程

DSI 调度程序线程 (DSI-S) 按提交顺序将小事务集中到组中。

在对事务分组后，DSI 调度程序就可以将组分派给下一个可用的 DSI 执行程序线程。DSI 调度程序会尝试以并行方式为不同源分派组，因为这些组可以并行提交。如果不同源的事务之间的争用很严重，请设置 **dsi_partitioning_rule** 参数的 **ignore_origin** 选项。

使用事务分区规则，您可以指定 DSI 调度程序可用于对事务进行分组的其它标准。

另请参见

- 分区规则：减少争用和增大并行度 （第 159 页）

DSI 执行程序线程

DSI 执行程序线程 (DSI-E) 将函数映射到函数字符串，然后在复制数据库中执行这些事务。

DSI 执行程序线程还会对复制数据服务器返回的任何错误采取相应操作。

使用并行 DSI 线程处理事务

可以使用 **dsi_large_xact_size** 数据库连接配置参数定义大事务和小事务。

dsi_large_xact_size 指定事务中允许包含的命令的数量，一旦超过此数值，此事务将被视为大事务。Replication Server 通常以不同方式处理小事务和大事务。

小事务

Replication Server 尝试将类似的事务进行分组，以便将它们作为一个较大的事务进行处理。

这样，Replication Server 就可以为组发出一次提交，而不是分别提交每个事务。如果满足以下任一个条件，事务组将完成并发送到下一个可用的 DSI 执行程序线程。例如：

- 已经从另一个源发出了下一个事务。
- 组中的事务数超过 **dsi_max_xacts_in_group** 指定的值。
- 组中事务的总大小（以字节为单位）超过 **dsi_xact_group_size** 指定的值。
- 下一个事务是大事务，大事务总是自己成为一个组。
- 事务分区规则决定下一个事务不能与现有的组组合。

在分组完成之后，就可以将它发送到下一个可用的 DSI 执行程序线程。只有已提交的事务才能添加到组中。也就是说，在读取事务的提交记录之后，才能将这些事务添加到事务组。

大事务

大事务会被提交到为大事务保留的下一个可用的 DSI 执行程序线程。

DSI 执行程序线程将事务发送到复制数据服务器，而无需等待获得提交记录。如果事务在主数据服务器上被回退，则 DSI 执行程序线程在复制数据服务器上将其回退。

如果 Replication Server 遇到大事务，而专用的大事务线程不可用，则按照处理小事务的方式处理该事务。

选择隔离级别

通过选择事务隔离级别，您可以控制事务执行期间数据可由其它用户访问的程度。

ANSI SQL 标准为事务定义了四种隔离级别。每种隔离级别都指定在处理并发事务时禁止的操作种类。较高级别包括由较低级别施加的限制。有关隔离级别的详细信息，请参见《Adaptive Server Enterprise Transact-SQL 指南》。

注意： Replication Server 不仅支持 ANSI 标准值，还支持需要复制到任何受支持的数据服务器的所有值。

- 级别 0 – 防止其它事务更改已由未提交的事务修改的数据。但是，其它事务仍可以读取未提交的数据，这会导致脏读。
- 级别 1 – 防止脏读，如果一个事务修改某一行，并且在此事务提交更改之前第二个事务读取该行，便会发生脏读。
- 级别 2 – 防止非重复读取，当一个事务读取某行而第二个事务对该行进行修改时，便发生非重复读取。如果第二个事务提交其更改，则第一个事务随后会读取到与原读取结果不同的结果。
- 级别 3 – 确保在某个事务结束之前其所读取的数据是有效的。它会通过在事务结束之前应用索引页或表锁来防止“非重复读取”和“幻像行”。

  如果要使用触发器强制实施数据库中数据的参照完整性，请选择隔离级别 3。隔离级别 3 可防止执行触发器时在表中出现幻像行。

您可以将 **create connection** 或 **configure connection** 与 **dsi_isolation_level** 选项一起使用来设置隔离级别。例如，要为与 SYDNEY_DS 数据服务器中 `pubs2` 数据库的连接将隔离级别更改为级别 3，请输入：

```
alter connection to SYDNEY_DS.pubs2
    set dsi_isolation_level to ’3’
```

Replication Server 使用 **rs_isolation_level** 系统变量将隔离级别值设置为 *rs_set_isolation_level* 函数字符串。在 Replication Server 与复制数据服务器建立连接后，**rs_set_isolation_level** 将会执行。如果未设置任何值，则 Replication Server 不会执行 **rs_dsi_isolation_level**，而是使用数据服务器的隔离级别。Adaptive Server 的缺省隔离级别为 1。

为非 Sybase 复制数据服务器设置隔离级别

隔离级别是可以变化的，具体取决于复制数据服务器。这会对在 Replication Server 中配置并行 DSI 产生影响。

可以为非 Sybase 复制数据服务器设置的隔离级别包括：

- Oracle – READ COMMITTED 和 SERIALIZABLE
- Microsoft SQL Server – READ UNCOMMITTED、READ COMMITTED、REPEATABLE READ、SNAPSHOT 和 SERIALIZABLE
- IBM DB2 UDB – REPEATABLE READ、READ STABILITY、CURSOR STABILITY 和 UNCOMMITED READ

必须为非 Sybase 复制数据服务器编辑 **rs_set_isolation_level** 函数字符串，并包括 *rs_isolation_level* 系统定义的变量。有关 **rs_set_isolation_level** 的详细信息，请参见《Replication Server 参考手册》。

如果您使用的是 Adaptive Server 之外的其它数据服务器，则当您修改数据服务器的 **rs_set_isolation_level** 函数字符串时，请确保包含 *rs_isolation_level* 变量。

要设置隔离级别，请在相应的函数字符串类中创建函数字符串。例如，在下面的语句中：

- Oracle – 要设置 SERIALIZABLE 隔离级别，请输入：

```
create function string rs_set_isolation_level
for rs_oracle_function_class
output language
'set transaction isolation level serializable'
```

- Microsoft SQL Server – 要设置 SERIALIZABLE 隔离级别，请输入：

```
create function string rs_set_isolation_level
for rs_msss_function_class
output language
'set transaction isolation level serializable
```

- IBM DB2 UDB – 要设置 REPEATABLE READ 隔离级别，请输入：

```
create function string rs_set_isolation_level
for rs_udb_function_class
output language
'set current isolation = RR'
```

事务序列化方法

Replication Server 提供了用于指定并行级别的不同序列化方法。

您选择的方法取决于您预期的并行线程之间的争用程度和复制环境。每种序列化方法都定义在必须等待上一个事务提交之前该事务可以开始的程度。

使用 **dsi_partitioning_rule** 参数可减少发生争用的可能性，而不会减少序列化方法指定的并行度。

序列化方法为：

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

使用 **alter connection** 命令及 **dsi_serialization_method** 参数，可以为数据库连接选择序列化方法。例如，输入以下命令，可为与 SYDNEY_DS 数据服务器上的 `pubs2` 数据库的连接选择 **wait_for_commit** 序列化方法：

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

事务包含三个部分：

- 开头
- 事务的主体，由 **insert**、**update** 或 **delete** 这样的操作组成
- 事务的结尾，由提交或回退组成

在提供提交一致性的同时，序列化方法将定义：事务是否要等到上一个事务准备好提交之后才开始，或者是否可以早一些开始处理事务。

另请参见

- 分区规则：减少争用和增大并行度 （第 159 页）

no_wait

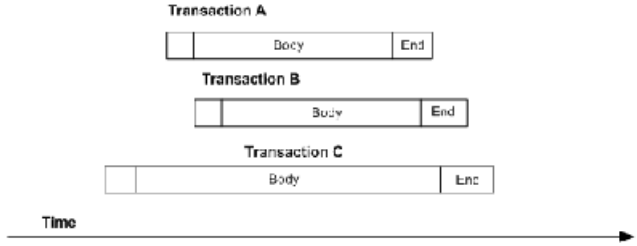
no_wait 方法指示 DSI 开始下一个事务，而不等待上一个事务提交。

此方法假定您的主应用程序设计为避免更新冲突，或者假定有效使用 **dsi_partitioning_rule** 来减少或消除争用。除非已将 **dsi_isolation_level** 设置为 **3**，否则 Adaptive Server 不持有更新锁。此方法假定并行事务之间几乎没有争用，并使执行几乎并行发生（如图中所示）。

no_wait 可以更好地提高性能，但同时也增加了产生争用的风险。

注意： 如果 **dsi_commit_control** 设置为“on”，则只能将 **dsi_serialization_method** 设置为 **no_wait**。

图 14： 使用 no_wait 序列化方法的线程计时

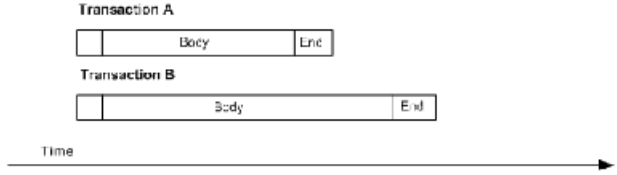


wait_for_start

wait_for_start 指定某个事务可以在预定在它紧前面提交的事务启动后立即启动。

Sybase 建议您不要在将 **dsi_serialization_method** 设置为 **wait_for_start** 的同时，将 **dsi_commit_control** 设置为 **off**。

图 15： 使用 wait_for_start 序列化方法的线程计时



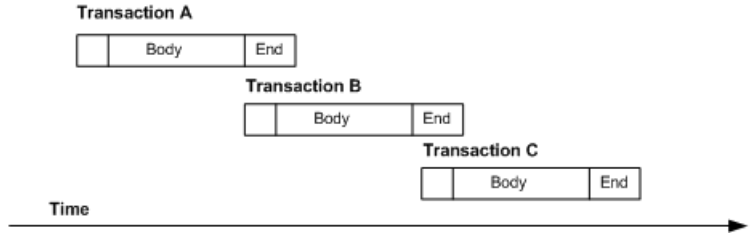
wait_for_commit

在 **wait_for_commit** 方法中，在成功处理上一个事务并发送提交之后，才发送下一个线程的事务组以进行处理。

这是缺省设置。它假定并行事务之间存在相当大的争用，并导致交错执行（如图所示）。

此方法指示 DSI 一直等到可以提交某个事务，然后再开始下一个事务，从而保持事务序列化。在提交第一个事务的同时，可以将下一个事务提交到复制数据服务器，因为第一个事务已经持有它所需的锁。

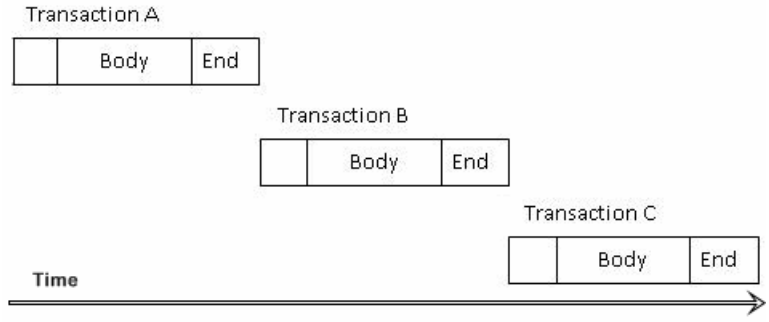
图 16： 使用 wait_for_commit 序列化方法的线程计时



wait_after_commit

wait_after_commit 指定一个事务要等到其前一个预定提交的事务已完全提交后才启动。

图 17： 使用 wait_after_commit 序列化方法的线程计时



分区规则：减少争用和增大并行度

使用 **dsi_partitioning_rule** 设置的分区规则，允许并行 DSI 功能根据具有公共名称、用户、重叠开始/提交时间或这些项的组合的事务，作出有关事务组和并行执行的决定。

分区规则允许并行 DSI 功能更加接近地模仿主数据服务器上的处理顺序，旨在用于减少复制数据服务器上的争用。

每个并行 DSI 参数都根据安装的条件提供了对功能进行微调的方法。**dsi_num_threads** 控制可用于连接的 DSI 线程数。**dsi_serialization_method** 控制连接的并行度，但是必须对增大的并行度与复制数据服务器上潜在的争用进行平衡。**dsi_partitioning_rule** 提供了一种减少争用的方法，但不会降低并行 DSI 功能的总体性能。

事务分区规则

Replication Server 允许您根据以下一个或多个属性，对每个连接的事务进行分区。

这些属性包括：

- 源
- 源和会话 ID
- none，不应用分区规则
- 用户名
- 原始开始时间和提交时间
- 事务名称
- 忽略源

注意： 如果要使用分区规则来提高性能，则 **dsi_serialization_method** 不得为 **wait_for_commit**。**wait_for_commit** 通过降低并行度来消除争用。

要选择分区规则，请使用带 **dsi_partitioning_rule** 选项的 **alter connection** 命令。语法为：

```
alter connection to data_server.database
    set dsi_partitioning_rule to '{ none|rule[, rule ] }'
```

rule 的值为 **user**、**time**、**origin**、**origin_sessid**、**name** 和 **ignore_origin**。例如，要根据用户名和原始开始时间和提交时间对事务进行分区，请输入：

```
alter connection to TOKYO_DS.pubs2
    set dsi_partitioning_rule to 'user,time'
```

分区规则：源

在来自同一源的事务应用于复制数据库时，**origin** 使这些事务序列化。

分区规则：源和进程 ID
在具有相同源和相同进程 ID 的事务应用于复制数据库时，**origin_sessid** 使这些事务序列化。

Sybase 建议首次尝试使用分区规则时，从 `origin_sessid,time` 设置开始。

注意： Application Server 的进程 ID 是会话进程 ID (SPID)。

分区规则：None
none 是缺省行为，此时的 DSI 调度程序将每个事务组或大事务指派给下一个可用的并行 DSI 线程。

分区规则：用户
如果您选择根据用户名对事务进行分区，则由同一主数据库用户 ID 输入的事务将被串行处理。只有由不同用户 ID 输入的事务才会被并行处理。

使用此分区规则可避免争用，但是在某些情况下可能会导致不必要的并行度损失。例如，某个 DBA 正在运行多个批处理作业。如果 DBA 使用同一用户 ID 提交每个批处理作业，Replication Server 会串行处理每个作业。

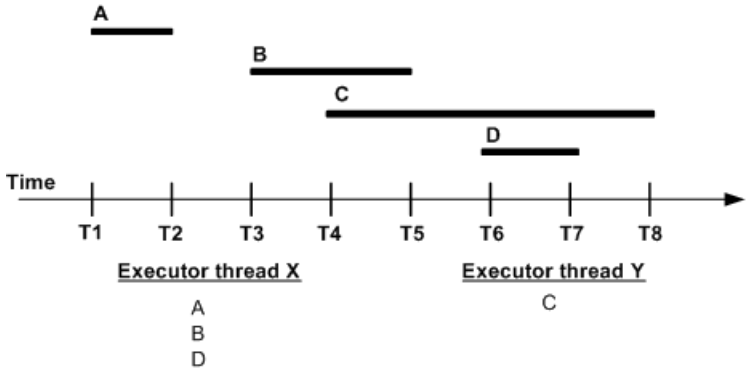
如果主数据库上的每个用户连接都具有唯一的 ID，则用户名分区规则是最有用的。如果多个用户使用同一 ID（如“sa”）登录，此分区规则就不太有用。在这类情况下，**orig_sessid** 可能是更好的选项。

分区规则：原始开始时间和提交时间
如果使用 **time** 分区规则，DSI 调度程序将查看事务的原始开始时间和提交时间，以确定主数据库上的同一进程不可能执行哪些事务。

如果事务的原始开始时间早于上一事务的提交时间，则该事务可以由不同的 DSI 执行程序线程处理。

假定已经选择了原始开始时间和提交时间分区规则，而且图中所示的事务和处理时间均来自同一主数据库。

图 18： 事务的原始开始时间和提交时间



在此示例中，DSI 调度程序将事务 A 指派给 DSI 执行程序线程 X。之后，调度程序比较事务 B 的开始时间和事务 A 的提交时间。由于事务 A 的提交时间早于事务 B 的开始时间，因此调度程序将事务 B 指派给执行程序线程 X。也就是说，事务 A 和 B 可以组合在一起，并可以由同一 DSI 执行程序线程处理。但是，事务 C 的开始时间早于事务 B 的提交时间。因此，调度程序假定事务 B 和 C 由主数据库上的不同进程应用，并将事务 C 指派给执行程序线程 Y。事务 B 和 C 不得在同一组中，可以用不同的 DSI 执行程序线程处理它们。因为事务 D 的开始时间早于事务 C 的提交时间，所以调度程序可以安全地将事务 D 指派给执行程序线程 X。

注意： 在处理大事务时，使用原始开始时间和提交时间分区规则可能会导致争用，因为总是在一个事务提交之前，调度另一个事务。

分区规则：名称

DSI 调度程序可以使用事务名称对事务分组，以进行串行处理。

在 Adaptive Server 上创建事务时，您可以使用 **begin transaction** 命令指派事务名称称。

如果应用事务名称分区规则，则 DSI 调度程序将同名的事务指派给同一执行程序线程。将并行处理具有不同事务名称的事务。**name** 参数将忽略名称为空或没有名称的事务。它们的处理过程取决于其它 DSI 并行处理参数或其它执行程序线程的可用性。

注意： 只有在非 Sybase 数据服务器支持事务名称时，此分区规则才可用于这些服务器。

缺省事务名称

缺省情况下，Adaptive Server 总是为每个事务指派名称。如果尚未使用 **begin transaction** 明确地指派名称，Adaptive Server 将指派一个名称，以下划线字符开头且包括说明事务的其它字符。例如，Adaptive Server 为单个 **insert** 命令指派缺省名称“_ins”。

将 **dsi_ignore_underscore_name** 选项与 **alter connection** 一起使用，可以指定 Replication Server 在根据事务名称对事务分区时是否忽略这些名称。缺省情况下，**dsi_ignore_underscore_name** 为 **on**，而且 Replication Server 处理名称以下划线开头的事务的方式与处理名称为空的事务的方式相同。

分区规则：忽略源

ignore_origin 会覆盖不同源事务的缺省处理规则，并允许将它们分区，就像它们全部同源一样。

除 **ignore_origin** 外，所有分区规则都允许并行应用来自不同源的事务，无论其它指定的分区规则如何。

例如：

```
alter connection dataserver.db
   set dsi_partitioning_rule to "name"
```

在这种情况下，将并行应用具有不同源的事务，无论它们是否具有相同名称。

name 分区规则只影响同源的事务。这样，即可以串行应用同源同名的事务，又可以并行应用同源不同名的事务。

如果 **ignore_origin** 在 **alter connection** 语句的最前面列出，Replication Server 则根据语句中的第二条或随后的规则对同源或不同源的事务进行分区。例如：

```
alter connection dataserver.db
   set dsi_partitioning_rule to "ignore_origin, name"
```

在这种情况下，即可以串行应用所有同名事务，又可以并行应用所有不同名的事务。事务的来源毫不相关。

如果 **ignore_origin** 在 **alter connection** 语句中的第二个或随后的位置列出，Replication Server 则忽略它。

使用多个事务规则

您可以为单个连接设置多个事务规则。

例如，同时应用原始会话 ID 以及原始开始时间和提交时间可以最接近于主数据库的处理环境。

当指定多个事务规则时，Replication Server 按规则在 **alter connection set dsi_partitioning_rule** 语法中的输入顺序来应用它们。

例如，如果将 **dsi_partitioning_rule** 设置为“time, user”，Replication Server 将在检查用户 ID 之前查看原始开始时间和提交时间。如果原始开始时间与提交时间之间不存在任何冲突，Replication Server 将检查用户 ID。如果原始开始时间与提交时间之间存在冲突，Replication Server 将应用 **time** 规则，而不会检查用户 ID。因此，如果后一个事务的原始开始时间早于提交时间，则会将两个事务指派给不同的并行 DSI 线程，即使这两个事务具有相同的用户 ID 也不例外。

分组逻辑和事务分区规则

分区规则可以影响分组及调度决策。

如果分区规则确定在重叠时间（**time** 规则）发生的两个事务具有不同的事务名称称（**name** 规则），或者它们来自不同的用户（**user** 规则），则不允许将这两个事务放在同一组中。否则，将根据事务大小、源等应用常规组大小决策。

另请参见

- 小事务 （第 154 页）

解决更新冲突

并行 DSI 处理必须按照事务在主数据库中提交的顺序进行，还允许同时处理事务更新。然后，它必须解决由此发生的任何事务争用。

当一个事务因必须等待较早的事务提交而无法提交，而较早的事务因所需资源被较晚的事务锁定也无法提交时，会提交顺序死锁事务争用（或争用死锁）。

例如，DSI 线程 A 和 B 并行处理事务。线程 A 的事务必须在线程 B 的事务之前提交。线程 B 的事务锁定线程 A 所需的资源。在线程 A 的事务提交之前，线程 B 的事务无法提交，但线程 A 的事务因所需资源被线程 B 锁定而无法提交。

Replication Server 提供了以下两种方法来解决提交顺序死锁：

- 在内部，使用 Replication Server 内部表和函数字符串，或者
- 在外部，使用复制数据库中的 `rs_threads` 系统表和一些函数字符串。

内部方法主要在 Replication Server 内进行处理，它使用 `rs_dsi_check_thread_lock` 函数字符串来检测提交顺序死锁。外部方法同时需要 Replication Server 和复制数据库，它使用 `rs_threads` 系统表验证提交顺序和检测提交顺序死锁。

Sybase 建议使用内部方法，对于 Sybase 和非 Sybase 数据服务器，该方法都是缺省方法。此方法比外部方法需要的网络 I/O 更少，而且在发生提交顺序死锁时它可能只需要回退一个事务。外部方法需要更多的网络 I/O，并且导致回退数个事务。包括外部方法是为了与早期版本的 Replication Server 兼容。

如果 Replication Server 遇到提交顺序死锁，且 **dsi_commit_control** 为 on，Replication Server 会回退并重试一个事务。如果 Replication Server 遇到提交顺序死锁且 **dsi_commit_control** 为 off，则 Replication Server 会以串行方式回退并重试所有事务。

要选择一种方法，请输入带 **dsi_commit_control** 选项的 **alter connection** 命令。例如，要为 TOKYO_DS 数据服务器上的 `pubs2` 数据库选择内部方法，请输入：

```
alter connection to TOKYO_DS.pubs2
set dsi_commit_control to 'on'
```

将 **dsi_commit_control** 设置为“on”可指定内部方法，将 **dsi_commit_control** 设置为“off”可指定外部方法。

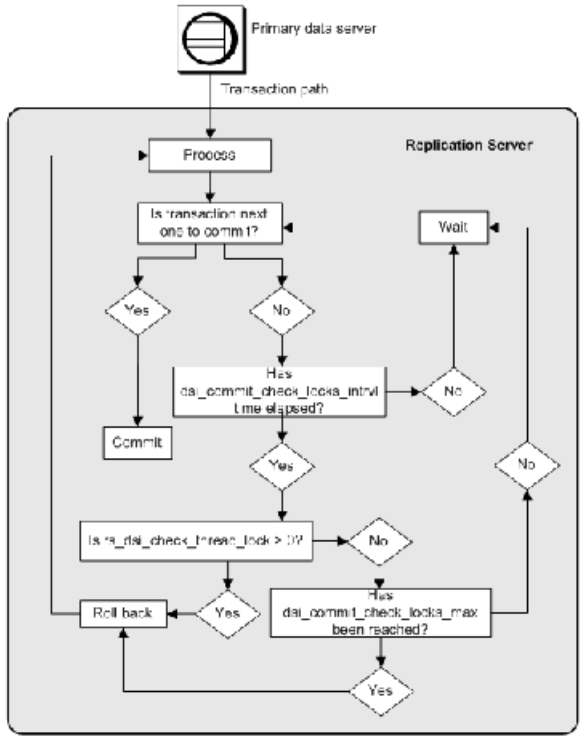
在内部解决冲突

了解 Replication Server 如何使用 **rs_dsi_check_thread_lock** 函数字符串在 Replication Server 中解决提交顺序死锁。

若要保持事务完整性，Replication Server 必须保持事务提交顺序并解决提交顺序一致性死锁。

此图介绍了 Replication Server 用于解决提交顺序死锁的逻辑。

图 19： 使用 rs_dsi_check_thread_lock 函数字符串解决冲突的逻辑



注意： 内部方法将解决 Replication Server 检测到的提交顺序死锁和仅发生在 Replication Server 内的更新冲突。如果复制数据库检测到死锁，该复制数据库将选择一个要回退的事务。为保证提交顺序，Replication Server 必须回退当前针对复制数据库执行的所有事务。之后，Replication Server 以串行方式重新应用这些事务。

保持提交顺序

Replication Server 读取从主数据库发送的提交信息，并使用此信息定义和保持复制数据库上的事务提交顺序。

如果 DSI 执行程序线程的事务已处理完毕，并且是应提交的“下一个”事务，则允许提交它。如果线程的事务已处理完毕，并且不是 应提交的“下一个”事务，则线程必须等待，直到轮到它时再提交事务。

解决提交一致性死锁
如果某个线程的事务已处理完毕，并且不是应提交的下一个事务，则该事务可能占用安排较早提交的事务所需的资源。等待 **dsi_commit_check_locks_intrvl** 参数中指定的时间后，DSI 执行程序线程执行 **rs_dsi_commit_check_thread_lock** 函数字符串，以确定线程是否持有较早事务所需资源上的锁：

- 如果线程阻塞另一个事务 (**rs_dsi_check_thread_lock** > 0)，则当前事务回退，这将解决提交顺序死锁问题并允许提交较早事务。仅回退阻塞的事务；正常处理其它事务。
- 如果线程没有阻塞另一个事务，它将检查执行 **rs_dsi_check_thread_lock** 的次数是否比 **dsi_commit_check_locks_max** 参数定义的次数多。
  - 如果线程执行 **rs_dsi_check_thread_lock** 的次数不比 **dsi_commit_check_locks_max** 中定义的次数多，则提交事务（如果它是下一个事务），或者再次等后 **dsi_commit_check_locks_intrvl** 指定的时间。
  - 如果线程执行 `rs_dsi_check_thread_lock` 的次数比 **dsi_commit_check_locks_max** 中定义的次数多，则回退当前事务。

用于内部提交控制的函数字符串
Replication Server 使用 **rs_dsi_check_thread_lock** 函数检查当前的 DSI 执行程序线程是否阻塞另一个复制数据库进程。

rs_dsi_check_thread_lock 确定 DSI 执行程序线程是否持有阻塞复制数据库进程的锁。如果返回值大于 0，表明线程占用另一个数据库进程所需的资源，而且此线程应该回退和重试事务。

此函数具有函数字符串类作用域。只有在 DSI 执行程序线程已经可以提交，但由于不是下一个要提交的事务而无法提交，而且为 **dsi_commit_check_locks_intrvl** 指定的时间已过时，才调用此函数。如果频繁发生提交顺序争用，请考虑减少 **dsi_commit_check_locks_intrvl** 指定的等待时间。

注意： Replication Server 自动在函数字符串类（Replication Server 在其中生成缺省函数字符串）中为上述函数创建函数字符串。对于其它函数字符串类，您必须先创建这些函数字符串，才能在 **dsi_commit_control** 设置为“on”的情况下使用并行 DSI 功能。

在外部解决冲突
了解 Replication Server 如何使用 `rs_threads` 表在外部解决提交顺序死锁。

`rs_threads` 表位于复制数据库中。每个 DSI 执行程序线程在其中都有对应的行。为模拟行级锁定，它包含两个列（`id` 和 `seq`）以及足够多的伪列，以使一页只有一行。`id` 列用作唯一聚簇索引。

在事务开始时，DSI 执行程序线程将用下一个可用的序列号对其在 rs_threads 表中的行进行更新。当可以提交事务时，线程将 **select** 语句发送到复制数据服务器，以便从 rs_threads 表中选择应该在此事务之前提交的事务的序列号。

由于上一个事务在 rs_threads 中的此行上持有锁，所以在上一个事务提交之前，将一直阻塞此线程。

如果返回的序列号比预期值小，线程会决定是应该回退事务还是重试 **select** 操作。由于 DSI 会先将许多命令设置为一个批处理的格式，然后再将它提交到 Adaptive Server，因此在上一个事务将任何命令提交到 Adaptive Server 之前，某个线程可能是可以提交的。如果是这种情况下，rs_threads 表中的 **select** 可能会被多次提交。

如果返回的序列号与预期值匹配，则可以提交事务。

解决死锁

了解 Replication Server 如何解决死锁。

如果一个事务可以提交，但由于它不是正常提交顺序中的下一个而无法提交，而且此事务持有必须在它之前提交的事务所需资源上的锁，则在复制数据库上将发生数据库资源死锁。

数据库资源死锁包括提交顺序中下一个事务持有的 rs_threads 上的锁，以及该事务所需资源上的锁。如果复制数据库检测到数据库资源死锁，将选择一个事务进行回退。

由于 Replication Server 必须保证提交顺序，因此当复制数据库强制实施此回退时，Replication Server 将回退针对复制数据库执行的所有事务，并按提交顺序以串行方式重新应用这些事务。

使用 rs_threads 控制提交的函数字符串

Replication Server 使用一些系统函数处理 rs_threads 系统表。

这些函数具有函数字符串类作用域。只有在为连接定义多个 DSI 线程时，才执行这些函数。

注意： 只有在将外部 rs_threads 方法用于提交控制时，才需要这些函数字符串。

表 22. 修改 rs_threads 系统表的系统函数

函数	说明
rs_initialize_threads	将 rs_threads 系统表中每个条目的序列设置为 0。此函数在初始化连接期间执行。
rs_update_threads	更新 rs_threads 系统表中指定条目的序列号。
rs_get_thread_seq	返回 rs_threads 系统表中指定条目的当前序列号。

函数	说明
rs_get_thread_seq_no-holdlock	使用 **noholdlock** 选项，返回 `rs_threads` 系统表中指定条目的当前序列号。当 **dsi_isolation_level** 为 3 时，将使用此线程。

配置并行 DSI 以优化性能

调优并行 DSI 处理以提供最佳复制性能，平衡并行处理与可接受的争用级别。

争用始终都会发生。消除争用的唯一方法是关闭并行 DSI 处理。

同时，为获得最大并行度而设置所有并行 DSI 参数可能会导致 Replication Server 从争用恢复的时间比将事务实际应用于复制数据库所用的时间还多。清楚地了解操作环境，以便您可以成功地平衡并行处理与可接受的争用级别，这样才能获取最佳性能。

准备配置并行 DSI 以优化性能

在开始性能调优之前，需要注意一些事项。

1. 了解您的事务配置文件。

   要复制哪些种类的事务？这些事务是否影响相同的行和表？如果并行应用这些事务，它们是否容易发生冲突？事务配置文件是保持不变，还是发生变化的（可能随日期或月份而变化）？清楚地了解您的事务配置文件有助于选择最有用的那些参数和设置。

2. 调优复制数据库以处理争用。

   通过使用聚簇索引、分区、行级锁定等，已对大多数主数据库进行了调优，以最大限度地减少争用。确保已经以类似的方式对复制数据库进行了调优。

3. 定义一组准确反映您的复制环境的可重复事务。

   调优并行 DSI 环境是一个迭代过程。您将需要设置参数、运行测试、测量性能、与以前的测量结果进行比较，并一直重复，直到获得最佳结果。

4. 重置 **dsi_serialization_method** 参数。

   注意： 如果 **dsi_commit_control** 设置为“on”，则只能将 **dsi_serialization_method** 设置为 **no_wait**。

   将 **dsi_serialization_method** 参数设置为 **no_wait** 以启用最大并行度。然后，尝试通过测试其它参数来减少争用。由于 **wait_for_commit**（缺省）设置提供了最小并行度，因而优点最少，因此只有在使用 **no_wait** 设置来减少争用的所有尝试都无法提高性能时才将 **dsi_serialization_method** 重新设置为 **wait_for_commit**。

5. 正确设置 **dsi_num_threads** 参数。

   dsi_num_threads 参数定义 DSI 执行程序线程的总数；**dsi_num_large_xact_threads** 参数定义为大事务保留的 DSI 执行程序线程总数。因此，DSI 执行程序线程 **(dsi_num_threads)** 的总数等于为大事务保留的 DSI 线程数加上可用于小事务的线程数。

若要开始，尝试将 **dsi_num_threads** 设置为 5，并将 **dsi_num_large_xact_threads** 设置为 2。在选择 **dsi_serialization_method** 和 **dsi_partitioning_rule** 之后：

- 如果争用没有增加，则增加 **dsi_num_threads**；或者，
- 如果争用没有减少，则减少 **dsi_num_threads**。

确保 **dsi_num_threads** 大于缺省值，并确保 **dsi_num_threads** 的值大于 **dsi_num_large_xact_threads** 的值。

减少争用

在完成准备配置并行 DSI 以优化性能任务，且性能测试指出争用正在影响性能时，开始调优并行 DSI 参数以减少争用。

例如：

- 复制正在阻止活动。
- 由于发生死锁，Replication Server 正在回退并重新应用大部分事务。请参见计数器 5060 – TrueCheckThrdLock。

从调优 **dsi_max_xacts_in_group** 参数开始，该参数确定在单个开始/提交块中分组的事务数。通过减小 **dsi_max_xacts_in_group** 的值，可以使 DSI 执行程序线程更频繁地提交。这样，DSI 执行程序线程在更短的时间段内占用更少的复制资源，从而减少争用。

调整 **dsi_num_threads** 参数也可以影响争用。可用的 DSI 执行程序线程数越大，在线程之间发生争用的可能性就越大。尝试将 **dsi_num_threads** 的值降至 3（其中一个线程是为大事务保留的）。查找提供最佳性能的值是一个迭代过程。请记住，如果总体性能提高，则一些争用是可接受的。

另请参见

- 准备配置并行 DSI 以优化性能 （第 167 页）

使用分区规则

分区规则也可以减少争用，但是需要您清楚地了解事务配置文件。

事务名称规则

查明事务是否具有事务名称，以及争用是否由具有相同名称的事务造成的。请尝试设置事务名称规则，该规则强制将同名事务逐个发送到复制数据库。

如果事务没有命名，则您可以更改应用程序以添加名称。然后，使用 **name** 规则仅对指定的事务进行序列化。假定 DSI 执行程序线程尝试并行处理两个或多个大事务时，某个特定类型的大事务始终产生问题。通过为有问题的事务指定相同的名称，并应用 **name** 规则，可以确保以串行方式处理有问题的事务。但是，请记住，**name** 规则应用于所有事务，所有同名事务都会被串行处理。

用户名规则
设置用户名规则可帮助您减少由于并行处理的事务具有相同用户 ID 而导致的争用。

与事务名规则类似，用户名规则适用于所有事务（如果设置），并串行处理具有相同用户 ID 的每个事务。

原始开始时间和提交时间规则
时间规则强制事务的串行执行，而且提交/开始时间不重叠。

也就是说，如果第一个事务的提交时间在下一个事务的开始时间之前，则必须串行执行这两个事务。

组合分区规则
您可以组合规则。要满足的第一个规则优先于其它规则。

因此，例如，如果指定了 **origin_sessid, time** 规则，则将强制具有相同原始会话 ID 的两个事务以串行方式运行，而不应用 **time** 规则。

频繁的更新冲突
如果事务间频繁发生冲突，请对并行 DSI 配置参数进行如下设置。

设置以下并行 DSI 配置参数：

- **dsi_serialization_method** – 将此参数设置为 **wait_for_commit**。
- **dsi_num_large_xact_threads** - 将此参数设置为 2。如果您要热备份应用程序中配置并行 DSI，请将备用数据库的 **dsi_num_larg_xact_threads** 参数设置为比在活动数据库中同时执行的大事务数大的值。
- **dsi_num_threads** – 将此参数设置为 3 加上 **dsi_num_large_xact_threads** 参数的值。如果您的事务通常较小（例如，包含一个或两个语句），请将 **dsi_num_threads** 设置为 1 加上 **dsi_num_large_xact_threads** 的值。

将 **parallel_dsi** 配置参数设置为“on”，可以提供配置并行 DSI 的快捷方法（如上所述）。它还将 **dsi_sqt_max_cache_size** 参数设置为 1 百万字节。

不频繁的更新冲突
如果事务间只是偶然发生冲突，请对并行 DSI 配置参数进行如下设置。

设置以下并行 DSI 配置参数：

- **dsi_isolation_level** – 如果复制数据服务器是 Adaptive Server，请将此参数设置为隔离级别 3。对于非 Sybase 数据服务器，请通过使用 **rs_set_isolation_level** 自定义函数字符串设置为与 ANSI 标准级别 3 对应的级别。
  - Oracle 和 Microsoft SQL Server – SERIALIZABLE 级别相当于 ANSI SQL 隔离级别 3。
  - DB2 – REPEATABLE READ 级别相当于 ANSI SQL 隔离级别 3。

- **dsi_num_large_xact_threads** - 将此参数设置为 2。如果您要热备份应用程序中配置并行 DSI，请将备用数据库的 **dsi_num_larg_xact_threads** 参数设置为比在活动数据库中同时执行的大事务数大的值。
- **dsi_num_threads** – 将此参数设置为 3 加上 **dsi_num_large_xact_threads** 参数的值。

另请参见

- 为非 Sybase 复制数据服务器设置隔离级别 （第 155 页）

设置隔离级别

使用 DSI 隔离级别可防止在启用并行 DSI 且将复制表配置为行级锁定时丢失部分事务。

在这些情况下，事务内各个操作的顺序与主数据库上所示的顺序也可能不匹配，即使事务本身是按照正确的顺序提交的也是如此。

例如，如果要提交的第二个事务更新了由要提交的第一个事务插入的行，则更新可能发生在提交之前。在这种情况下，可以正确提交事务，但是更新将丢失，即使已保留插入操作也是如此。

若要避免顺序混乱的 DML 操作，请将 **dsi_isolation_level** 设置为 3。在此示例中，如果 **dsi_isolation_level** 为 **3**，则要提交的第二个事务会要求对其打算更新的尚不存在的行进行范围锁定，这将造成要提交的第一个事务死锁。数据服务器将声明数据库资源死锁。Replication Server 会回退所有打开的事务并以串行方式重新应用它们，而更新不会丢失。

设置大事务的大小

将 **dsi_large_xact_size** 设置为一个较大的数字，甚至是最大值 (2,147,483,647)，以免除处理大事务所需的开销，从而提供比允许大事务在其提交点到达之前启动更好的性能。

Parallel DSI and the rs_origin_commit_time 系统变量

rs_origin_commit_time 系统变量的值取决于您是否使用并行 DSI 功能。

- 如果没有使用并行 DSI 处理大事务，则 *rs_origin_commit_time* 值包含事务组中最后一个事务在主节点上的提交时间。
- 如果使用并行 DSI 处理大事务（在从 DSI 队列读取其提交之前），当 DSI 线程开始处理其中一个事务时，*rs_origin_commit_time* 值将设置为 *rs_origin_begin_time* 值。

  读取事务的提交语句时，*rs_origin_commit_time* 的值将被设置为实际提交时间。因此，如果将配置参数 **dsi_num_large_xact_threads** 设置为大于零的值，则 *rs_origin_commit_time* 的值对除 **rs_commit** 以外的所有系统函数都不可靠。

DSI 批量拷入

Replication Server 支持批量拷入，这可在复制复制数据库中的相同表上的大批量 **insert** 时提高性能。

在正常复制过程中，当复制到复制数据库时，Replication Server 将组成 SQL **insert** 命令，将该命令发送到复制数据库，然后等待复制数据库处理该行并发回操作结果。当复制大批量数据（如营业日结束时的批处理或交易合并）时，这会影响 Replication Server 的性能。

数据库支持

Adaptive Server 数据库以及由 ExpressConnect for Oracle 更新的 Oracle 复制数据库支持批量拷入。如果打开 DSI 批量拷入功能并且不支持复制数据库，DSI 将出错并关闭。请参见“Replication Server Options”的《ExpressConnect for Oracle 安装和配置指南》中的“系统要求”。

DSI 批量拷入配置参数

控制 DSI 中的批量操作的数据库连接参数。

参数	说明
dsi_bulk_copy	为连接打开或关闭批量拷入功能。如果 **dynamic_sql** 和 **dsi_bulk_copy** 均设置为 on，Replication Server 会在适当时应用批量拷入，如果 Replication Server 无法使用批量拷入，则使用动态 SQL。 缺省值：off。
dsi_bulk_thresh-old	事务中的连续 **insert** 命令数，在达到此数字时，将会触发 Replication Server 使用批量拷入。当稳定队列事务 (SQT) 遇到大批量 **insert** 命令时，它将在内存中保留指定数量的 **insert** 命令以确定是否应用批量拷入。由于这些命令保留在内存中，Sybase 建议您不要将该值配置为比 **dsi_large_xact_size** 配置值高太多。 最小值：1 缺省值：20

设置批量拷入

可以使用 **alter connection** 或 **configure replication server** 设置批量拷入参数值。

使用：

- **alter connection** 用于在连接级别更改批量拷入连接参数：

```
alter connection to dataserver.database
set {dsi_bulk_copy | dsi_bulk_threshold} to value
```

- **configure replication server** 用于更改服务器缺省值：

```
configure replication server
set {dsi_bulk_copy | dsi_bulk_threshold} to value
```

要检查 **dsi_bulk_copy** 和 **dsi_bulk_threshold** 的值，请使用 **admin config**。

当 **dsi_bulk_copy** 为 on 时，SQT 会计算事务所包含的同一表中连续 **insert** 语句的数量。如果该数量达到 **dsi_bulk_threshold**，DSI 会：

1. 将数据批量拷入到 Adaptive Server，直至 DSI 到达不是 **insert** 的命令或属于不同复制表的命令。
2. 继续执行事务中的其余命令。

Adaptive Server 将在批量拷入操作结束时（如果操作成功）或在故障点处发送批量操作结果。

注意： 批量拷入的 DSI 实现支持多语句事务，从而允许 DSI 执行批量拷入，即使事务中包含不属于批量复制的命令。

对预订实现的更改

批量拷入改善了预订实现的性能。

如果将 **dsi_bulk_copy** 设置为 on 并且每个事务中的 **insert** 命令数超过 **dsi_bulk_threshold**，Replication Server 将使用批量拷入实现预订。

注意： 在正常复制过程中，如果将 **autocorrection** 设置为 on，则会为表禁用批量操作。不过，如果达到了 **dsi_bulk_threshold** 并且实现不是从故障中恢复的非原子预订，则会在实现过程中应用批量操作，即使启用了 **autocorrection** 也不例外。

有关预订实现方法的信息，请参见《Replication Server 管理指南第一卷》中的“管理预订”。

用于批量拷入的计数器

支持批量拷入的计数器。

计数器	说明
DSINoBulkDatatype	由于数据包含与批量拷入不兼容的数据类型而跳过的批量操作数量。
DSINoBulkFstr	由于表自定义了 **rs_insert** 或 **rs_writetext** 函数字符串而跳过的批量操作数。
DSINoBulkAutoc	由于表启用了 **autocorrection** 而跳过的批量操作数。
DSIEBFBulkNext	由于下一个命令是批量拷入而执行的批处理刷新数。
DSIEBulkSucceed	数据服务器接口执行程序 (DSI/E) 在目标数据库中调用 **blk_done(CS_BLK_ALL)** 的次数。
DSIEBulkCancel	DSI/E 在目标数据库中调用 **blk_done(CS_BLK_CANCEL)** 的次数。

计数器	说明
DSIEBulkRows	DSI/E 使用批量拷入向复制数据服务器发送的行数。
BulkTime	DSI/E 花费在使用批量拷入向复制数据服务器发送数据的时间量（以毫秒为单位）。

批量拷入限制

在使用批量拷入时，应注意一些限制。

在以下情况下，Replication Server DSI 不使用批量拷入：

- Autocorrection 为 on 且数据不是预订实现的一部分。
- **rs_insert** 具有用户定义的函数字符串。
- `text` 列具有用户定义的 **rs_writetext** 函数字符串且输出为 `none` 或 `rpc`。
- 数据行包含 `opaque` 数据类型或用户定义的数据类型 (UDD)，且该数据类型的 `rs_datatype.canonic_type` 值为 255。

在以下列出的情况下，不支持批量拷入功能。在这些情况下，禁用批量拷入。

- 复制数据库不支持批量拷入。在这种情况下，如果启用了 DSI 批量拷入，DSI 将终止并显示一条错误消息。请参见《Replication Server 异构复制指南》。
- 数据大小在 Replication Server 字符集和复制 Adaptive Server 字符集之间发生改变，并且数据行包含文本列。在这种情况下，如果启用 DSI 批量拷入，DSI 将终止并显示此错误消息：

  ```
  Bulk-Lib routine 'blk_textxfer' failed.
  Open Client Client-Library error: Error: 16843015,
  Severity 1 -- 'blk_textxfer(): blk layer: user
  error: The given buffer of xxx bytes exceeds the
  total length of  the value to be transferred.'
  ```
- `owner.tablename` 长度大于 255 字节，并且复制数据库早于 15.0.3 过渡版本。如果启用 DSI 批量拷入，Replication Server 将终止并显示此消息：

  ```
  Bulk-Lib routine 'blk_init' failed.
  ```

  要指定当 `owner.tablename` 长度大于 255 字节时不使用批量拷入，请输入：

  1. 打开跟踪：

     ```
     trace "on", rsfeature, ase_cr543639
     ```
  2. 将其添加到 Replication Server 配置文件中：

     ```
     trace=rsfeature,ase_cr543639
     ```

其它限制：

- 与 **insert** 命令不同，批量拷入不生成时间戳；如果复制中不包含 `timestamp` 列，则会在 `timestamp` 列中插入 NULL 值。请禁用批量拷入，或者将复制定义设置为包含 `timestamp` 列。
- 即使将 **writetext** 函数字符串更改为 **no log**，也会始终记录 `Text` 和 `image` 列。

- 在 Adaptive Server 中，批量拷入不调用 **insert** 触发器。
- **send_timestamp_to_standby** 配置参数在批量拷入中无效。始终会将 `timestamp` 数据复制到备用数据库。

SQL 语句复制

Replication Server 在 Adaptive Server 中支持 SQL 语句复制，这可以补充基于日志的复制，并解决批量作业导致的性能降低问题。

Sybase 建议您在以下情况下使用 SQL 语句复制：

- DML（数据操作语言）语句影响复制表上的很多行。
- 更改基础应用程序以便启用存储过程复制时遇到困难。

注意： 日志传送管理器不支持 SQL 语句复制并且不支持将 SQL 语句复制到非 Adaptive Server 数据库。

SQL 语句复制简介

在 SQL 语句复制中，Replication Server 收到的是已修改主数据的 SQL 语句，而不是事务日志的单行更改。

Replication Server 将 SQL 语句应用于复制节点。Adaptive Server RepAgent 发送 SQL 数据操作语言 (DML) 和单行更改。根据您的配置，Replication Server 选择单行更改日志复制或 SQL 语句复制。

SQL 语句复制包括行计数验证，以确保复制后主数据库与复制数据库中更改的行数相符。如果行数不相符，您可以指定 Replication Server 处理此错误的方式。

要启用和配置 SQL 语句复制，请执行以下操作：

- 配置主数据库以记录 SQLDML。
- 配置 Replication Server 以复制 SQLDML：
  1. 使用 SQLDML 为表和多节点可用性 (MSA) 复制创建复制定义。
  2. 在 Replication Server 中，将热备份复制的 **WS_SQLDML_REPLICATION** 参数设置为 on。

基于日志的复制的性能问题

了解基于日志的复制如何影响性能以及如何解决性能问题。

Sybase 复制技术基于日志。在复制表上执行的修改将记录到数据库事务日志中。Adaptive Server 为对每个受影响的行的每处修改生成一个日志记录；单个 DML 语句可能会导致 Adaptive Server 生成多个日志记录。根据 DML 语句类型，Adaptive Server 可能会为每个受影响的行记录一个“before”image 和一个“after”image。Sybase Replication Agent 读取日志并将其转发到 Replication Server。Replication Server 确定 DML 操作（**insert**、**delete**、**update**、**insert**、**select** 或存储过程执行），并为每个操作生成相应的 SQL 语句。

基于日志的复制具有以下固有问题：

- 当单个 DML 语句影响多个行时，Replication Server 会在复制节点上应用多个 DML 语句，而不只是单个原始 DML 语句。例如，如果复制 `t` 表，则：

  ```
  1> delete tbl where c < 4
  2> go
  (3 rows affected)
  ```

  delete 语句在事务日志中记录三个记录，每个删除的行对应一个记录。这些日志记录将用于数据库恢复和复制。Replication Agent 将与三个日志记录有关的信息发送到 Replication Server，后者将该信息重新转换为三个 **delete** 语句：

  ```
  delete t where c = 1
  delete t where c = 2
  delete t where c = 3
  ```
- Adaptive Server 无法在导致在复制数据库上非对称装载资源的复制节点上执行优化。
- 处理影响多个行的大量语句将会增加系统中的延迟。
- Adaptive Server 只部分记录有关 **select into** 的信息；因此，复制系统无法成功复制 DML 命令。

可以通过两种不同的方法来解决所有以下问题：

- 存储过程复制
- SQL 语句复制

存储过程复制

您可以使用存储过程复制封装复杂的 DML 操作或影响很多行的操作。

通过仅复制存储过程调用而忽略对各个行的修改，存储过程复制可以提高性能。将降低网络通信量，Replication Server 需要较少的处理即可在复制节点中应用存储过程。

在复制 DDL 的热备份配置中，无法复制 **select into** 操作，因为这些操作进行了最少的记录。由于复制处理和 **select into** 命令固有的事务管理限制，无法使用存储过程复制。

此外，无法方便修改某些第三方应用程序以支持存储过程复制。因此，即使存储过程复制提高了 Replication Server 性能，也不能在任何情况下都使用它。

Replication Server 拓扑如何影响 SQL 语句复制

要使用 SQL 语句复制，您必须考虑到基础 Replication Server 拓扑。

Replication Server 支持多种不同的拓扑结构，包括“基本主复制”模型，其中可能包括几个 Replication Server、热备份配置以及多节点可用性 (MSA) 配置。

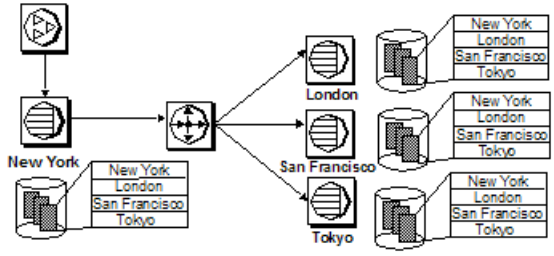
与传统的复制一样，SQL 语句复制基于日志；复制 SQL 语句（在主数据库中执行）所需的信息存储在事务日志中。日志读取器、Sybase Replication Agent 或其它应用程序将读取事务日志以通知 Replication Server 对复制表的修改。

在简单的 MSA 或热备份配置中，源和目标数据是相同的，在主表上执行的 DML 语句影响复制表上的相同数据集。

注意： SQL 语句复制仅适用于 DML 语句。

复制节点中具有相同的数据
此图显示了在纽约具有单个主数据库的 Replication Server 拓扑。表将复制到三个其它节点：伦敦、东京和旧金山。将完全复制所有表。

图 20： 基本主复制模型：复制节点中具有相同的数据



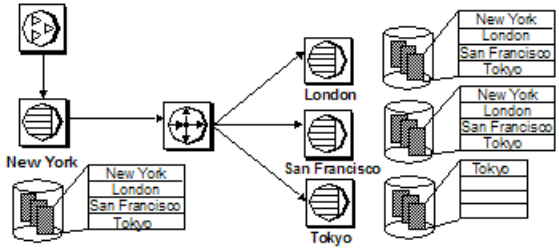
请考虑连接到纽约节点的客户端执行的以下语句：

```
delete t1 where a>5
```

如果在东京、伦敦和旧金山执行此命令，则影响会所有复制节点中具有相同的数据集，因为所有节点中的数据是相同的。在这种情况下，可以将所有复制节点配置为使用 SQL 语句复制。

复制节点中具有不同的数据
在此图表示的系统中，东京的复制节点仅预订 `site` 等于“Tokyo”的数据子集。

图 21： 基本主复制模型：复制节点中具有不同的数据



请考虑在纽约节点中执行的以下语句：

```
delete t1 where a>5
```

Replication Server 可以在伦敦和旧金山执行相同的语句，但不能在东京执行该语句，因为此节点仅预订一个数据子集。如果在这种情况下使用 SQL 语句复制，某些复制型数据库（如东京节点）将根据传统复制从主事务日志中接收各个日志记录修改。其它复制型数据库（如伦敦节点）将收到 SQL 语句。

如果主数据库和复制数据库具有不同的对象模式，或者用户使用与另一个表的 **join** 来执行 DML 语句，则可能会影响主表和复制表上的不同数据集。在这些情况下，将影响主数据库和复制数据库上的不同数据。用于 **join** 的表可能未标记为要复制，或者该表中的值可能是不完整的或不同于主数据库。

您必须在保存主数据的 Adaptive Server 以及 Replication Server 中激活 SQL 语句复制。在主 Adaptive Server 上启用 SQL 语句复制后，Adaptive Server 将在事务日志中为每个执行的激活了 SQL 语句复制的 DML 语句记录额外的信息。Replication Agent 或其它日志读取器为复制到 Replication Server 的 SQL 语句提供各个日志记录修改和信息。

注意： 对于 Replication Server 15.2 和更高版本，Sybase Replication Agent 将发送 SQL 语句复制信息。

如果在复制节点上应用的语句可能会影响不同的数据集，则 Adaptive Server 禁止使用 SQL 语句复制。

启用 SQL 语句复制

您可以在数据库、表或会话级别启用 SQL 语句复制。会话设置将覆盖表和数据库设置。表设置将覆盖数据库设置。

有几种 Adaptive Server 存储过程支持 SQL 语句复制。

在数据库级启用 SQL 语句复制

可以使用 **sp_setrepdbmode** 在数据库级为特定 DML 操作启用 SQL 语句复制。

适用于 SQL 语句复制的 DML 操作包括：

- **U – update**
- **D – delete**
- **I – insert select**
- **S – select into**

例如，要将 **delete** 语句作为 SQL 语句复制，同时启用 **select into** 复制，请输入：

```
sp_setrepdbmode pdb, 'DS', 'on'
```

当用户在 pdb 数据库的表上执行 **delete** 时，Adaptive Server 将记录有关 SQL 语句复制的额外信息。RepAgent 将发送各个日志记录和 Replication Server 所需的信息以构建 SQL 语句。只有在将数据库标记为要进行复制时（通过将 **sp_setreptostandby** 设置为 **ALL** 或 **L1**），您才能在数据库级设置 SQL 语句复制。

threshold 参数定义 DML 语句必须影响的最小行数，达到此行数后才能激活 SQL 语句复制。缺省阈值为 50 行，这表示如果 DML 语句至少影响 51 行，Adaptive Server 自动使用 SQL 语句复制。

例如，要在数据库级设置阈值，以便当数据操作语言 (DML) 语句影响 100 行以上时触发 SQL 语句复制，请输入：

```
sp_setrepdbmode pubs2, ‘threshold’, ‘100’
go
```

请参见《Replication Server 参考手册》的“Adaptive Server 命令和系统过程”中的“**sp_setrepdbmode**”。

另请参见

- 设置 SQL 语句复制阈值 （第 180 页）

显示 SQL 语句复制状态

可以使用 **sp_reptostandby** 显示数据库级别的 SQL 语句复制状态。

例如：

```
sp_reptostandby pdb
go
The replication status for database 'pdb' is 'ALL'.
The replication mode for database 'pdb' is 'off'.
```

在表级启用 SQL 语句复制

可以使用 **sp_setrepdefmode** 在表级配置 SQL 语句复制。表级设置将覆盖数据库级设置。

sp_setrepdefmode 包括执行以下操作的选项：

- 为特定 DML 操作启用或禁用 SQL 语句复制
- 配置激活 SQL 语句复制必须达到的阈值

适用于 SQL 语句复制的 DML 操作包括：

- **U – update**
- **D – delete**
- **I – insert select**

例如，要为 t 表上的 **update**、**delete** 和 **insert select** 操作启用 SQL 语句复制，请使用：

```
sp_setrepdefmode t, 'UDI', 'on'
go
```

当用户在 t 表上执行 **delete**、**update** 或 **insert select** DML 语句时，Adaptive Server 将记录有关 SQL 语句复制的额外信息。RepAgent 读取日志并发送各个日志记录和 Replication Server 所需的信息以构建 SQL 语句。

threshold 参数定义 DML 语句必须影响的最小行数，达到此行数后才能激活 SQL 语句复制。缺省阈值为 50 行，这表示如果 DML 语句至少影响 51 行，Adaptive Server 自动使用 SQL 语句复制。

例如，要将阈值设置为 100，请使用：

```
sp_setreptable t, true
go
sp_setrepdefmode t, 'UD', 'on'
go
sp_setrepdefmode t, 'threshold','100'
go
```

在此示例中，如果在 t 表上执行的 **update** 和 **delete** 语句至少影响 101 行，则这些语句使用 SQL 语句复制。

请参见《Replication Server 参考手册》的“Adaptive Server 命令和系统过程”中的“**sp_setrepdefmode**”。

注意： 无法在表级配置 **select into** 操作，因为目标表尚不存在。

另请参见

- 设置 SQL 语句复制阈值 （第 180 页）

在会话级启用 SQL 语句复制

在会话期间，可以使用 **set repmode** 为指定的 DML 操作配置 SQL 语句复制。会话设置将覆盖数据库级和对象级设置。

可以在登录时（使用登录触发器）或批处理开始时指定会话级设置。

例如，要在会话期间仅将 **select into** 和 **delete** 作为 SQL 语句复制，请使用：

```
set repmode on 'DS'
```

可以使用 **set repmode off** 删除会话级的所有 SQL 语句复制设置。

在会话期间，**set** 操作将处于活动状态。在存储过程完成后，在存储过程内设置的选项将恢复为缺省值。

注意： 在登录触发器内设置选项时，将在触发器完成执行后保留这些选项设置。

只有在设置了会话级选项 **set replication on** 时，执行 **set repmode on** 才会启用 SQL 语句复制。此示例不启用 SQL 语句复制：

```
set replication off
go
set repmode on 'S'
go
```

此示例启用 SQL 语句复制：

```
sp_reptostandby pdb, 'ALL'
go
```

```
set repmode on 'S'
go
```

threshold 参数定义 DML 语句必须影响的最小行数，达到此行数后才能激活 SQL 语句复制。缺省阈值为 50 行，这表示如果 DML 语句至少影响 51 行，Adaptive Server 自动使用 SQL 语句复制。

此示例显示如何在会话级将阈值定义为 1000 行：

```
set repmode ‘threshold’, ‘1000’
go
```

请参见《Replication Server 参考手册》的“Adaptive Server 命令和系统过程”中的“**set repmode**”。

另请参见

- 设置 SQL 语句复制阈值 （第 180 页）

设置 SQL 语句复制阈值

您可以引发 SQL 语句复制，而不必在各个表上设置阈值。

您可以在以下级别设置阈值：

- 数据库级别 – 使用 Adaptive Server 15.0.3 ESD #1 和更高版本。
- 会话级别 – 使用 Adaptive Server 15.0.3 ESD #2 和更高版本。

在 Adaptive Server 15.0.3 中，您只能在表级设置阈值。

缺省情况下，在 SQL 语句影响超过 50 行时，将引发 SQL 语句复制。您可以在会话、数据库和表级别设置不同的阈值。

不过，在会话级设置的阈值覆盖表级和数据库级的阈值，并且为任何表设置的阈值覆盖在数据库级设置的阈值。

在数据库级设置阈值和操作

可以使用 **sp_setrepdbmode** 命令的 **threshold** 参数在数据库级设置阈值。

这些示例显示如何在数据库和表级别设置阈值，同时显示如何在不同的级别定义操作。

示例 1

此示例显示在数据库级和表级为 `pubs2` 数据库和 `table1` 表设置不同的阈值：

1. 在数据库级将阈值重置为缺省值 50 行：

   ```
   sp_setrepdbmode pubs2, ‘threshold’, ‘0’
   go
   ```

2. 启用 `pubs2` 的 **update**、**delete**、**insert** 和 **select into** 操作的 SQL 语句复制：

   ```
   sp_setrepdbmode pubs2, ‘udis’, ‘on’
   go
   ```

3. 只有在步骤 2 中定义的操作影响超过 1,000 行时，才会为这些操作引发 pubs2 中的 table1 的 SQL 语句复制：

```
sp_setrepdefmode table1, ‘threshold’, ‘1000’
go
```

示例 2

此示例显示如何在数据库级为 pubs2 定义阈值，同时为位于 pubs2 数据库中的 table1 和 table2 等表定义不同的操作：

1. 在数据库级设置阈值，以便当数据操作语言 (DML) 语句影响 100 行以上时触发 SQL 语句复制：

```
sp_setrepdbmode pubs2, ‘threshold’, ‘100’
go
```

2. 为两个特定表定义一组不同的操作，并在其中使用 SQL 语句复制来复制操作。update、delete 和 insert 操作针对的是 table1，delete 操作针对的是 table2：

```
sp_setrepdefmode table1, ‘udi’, ‘on’
go
sp_setrepdefmode table2, ‘d’ ‘on’
go
```

在对 table2 执行 **delete** 操作时，或在 table1 上执行任何 DML 时，如果达到您在数据库级指定的阈值 100 行，将会触发 SQL 语句复制。

在会话级设置阈值和操作

可以使用 **set repthreshold** 在会话级设置阈值。

在会话级定义的阈值将覆盖在表或数据库级设置的阈值。在表级设置的阈值将覆盖在数据库级设置的阈值。

这些示例显示如何在会话、数据库和表级设置阈值，同时显示如何在不同的级别定义操作。

示例 1

此示例显示如何在未在数据库级和表级设置任何阈值的情况下在会话级将阈值定义为 23，或者覆盖表级和数据库级的阈值设置：

```
set repthreshold 23
go
```

示例 2

此示例显示如何在会话级将阈值重置为缺省值 50：

```
set repthreshold 0
go
```

示例 3

此示例显示如何在数据库和表级为 pubs2 数据库和 table1 表设置不同的阈值，然后仅为该会话定义一个不同的操作：

1. 在数据库级将阈值重置为缺省值 50 行：

```
sp_setrepdbmod pubs2, ‘threshold’, ‘0’
go
```

2. 启用 pubs2 的 **update**、**delete**、**insert** 和 **select into** 操作的 SQL 语句复制：

```
sp_setrepdbmode pubs2, ‘udis’, ‘on’
go
```

3. 只有在 DML 操作影响超过 1,000 行时，才会为 pubs2 中的 table1 引发 SQL 语句复制：

```
sp_setrepdefmode table1, ‘threshold’, ‘1000’
go
```

4. 仅为任何表上的 **update** 操作启用 SQL 语句复制以及仅为此会话启用 SQL 语句复制。这将覆盖步骤 2 中的数据库级设置：

```
set repmode on ‘u’
go
```

示例 4

您可以在 Adaptive Server 存储过程内调用 **set reptheshold**。此示例显示如何创建 **set_rep_threshold_23** 存储过程并在 **my_proc** 存储过程中进行调用：

1. 创建 **set_rep_threshold_23** 存储过程：

```
create procedure set_rep_threshold_23
as
set repthreshold 23
update my_table set my_col = 2  (statement 2)
go
```

2. 创建 **my_proc** 存储过程：

```
create procedure my_proc
as
update my_table set my_col = 1   (statement 1)
exec set_rep_threshold_23
update my_table set my_col = 3   (statement 3)
go
```

3. 执行 **my_proc** 以调用 **set_repthreshold_23**：

```
exec my_proc
go
```

在 **my_proc** 存储过程内，首先执行语句 1，阈值为 50。然后执行语句 2，阈值为 23。接下来执行语句 3，阈值为 50，这是因为 **set repthreshold 23** 命令仅在执行 **set_rep_threshold_23** 过程时有效。

示例 5

会话级阈值是可以导出的。因此，您可以将某个过程的 **export_options** 设置设为“on”，然后设置 SQL 语句复制阈值，使外部作用域中的过程使用由该存储过程设置的 SQL 语句复制阈值：

1. 创建 **set_repthreshold_23** 存储过程并将 **export_options** 设置为 on：

```
create procedure set_repthreshold_23
as
```

```
set repthreshold 23              (statement 4)
set export_options on
update my_table set my_col = 2  (statement 2)
go
```

2. 创建 **my_proc** 存储过程：

```
create procedure my_proc
as
update my_table set my_col = 1   (statement 1)
exec set_rep_threshold_23
update my_table set my_col = 3   (statement 3)
go
```

3. 执行 **my_proc** 以调用 **set_repthreshold_23**：

```
exec my_proc
go
```

首先执行语句 1，阈值为 50。然后执行语句 2，阈值为 23。接下来执行语句 3，阈值为 23，这是因为 **set repthreshold 23** 命令的作用域是会话的作用域。

示例 6

您可以创建登录触发器，以便自动为特定的登录 ID 设置复制阈值。

1. 创建阈值设置为 23 的 **threshold** 存储过程并启用导出：

```
create proc threshold
as
set repthreshold 23
set export_options on
go
```

2. 指示 Adaptive Server 当用户“Bob”登录时自动运行 **threshold** 存储过程：

```
sp_modifylogin Bob, 'login script', threshold
go
```

当 Bob 登录到 Adaptive Server 时，会话的 SQL 语句复制阈值设置为 23。

设置阈值并配置复制

您可以使用未配置为要复制的数据库，同时在数据库级为 SQL 语句复制设置阈值。

例如：

```
sp_reptostandby pubs2, 'none'
go
sp_setrepdbmode pubs2, 'threshold', '23'
go
```

不过，要在数据库级定义操作，您还必须在数据库级配置复制。例如，您不能执行：

```
sp_reptostandby pubs2, 'none'
go
sp_setrepdbmode pubs2, 'udis', 'on'
go
```

为 SQL 语句复制配置复制定义

您可以在数据库和表级为复制定义更改 SQL 语句复制选项。

数据库复制定义

要在 MSA 环境中复制 SQL 语句，您必须在 **create database replication definition** 或 **alter database replication definition** 命令中包括 **replicate SQLDML** 子句。

create database replication definition 或 **alter database replication definition** 语法必须包括以下子句：

```
[[not] replicate setname [in (table list)] ]
```

其中：

setname = **DDL** | **tables** | **functions** | **transactions** | **system procedures** | **SQLDML** | **‘options’**。

‘options’ 参数是以下语句的组合：

- **U** – update
- **D** – delete
- **I** – insert select
- **S** – select into

SQLDML 参数也定义为 **U**、**D**、**I** 和 **S** 语句的组合。

以下示例说明如何使用 **‘options’** 参数将 SQLDML 复制到 `tb1` 和 `tb2` 表。

```
replicate 'UDIS' in (tb1,tb2)
```

以下示例说明如何使用 **SQLDML** 参数，它可以生成与上一示例中的 **‘options’** 参数相同的结果：

```
replicate SQLDML in (tb1,tb2)
```

您可以在 **create database replication definition** 中使用多个 replicate 子句。但对于 **alter database replication definition**，只能使用一个子句。

如果在复制定义中未指定过滤器，则缺省过滤器为 **not replicate** 子句。若要更改 SQLDML 过滤器，请应用 **alter database replication definition**。您可以在 **replicate** 子句中指定一个或多个 SQLDML 过滤器。

以下示例说明如何为所有表过滤掉 **select into** 语句。第二个子句 (**not replicate 'U' in (T)**) 将过滤掉对 `T` 表的更新：

```
create database replication definition dbrepdef
       with primary at ds1.pdb1
       not replicate ‘S’
       not replicate ‘U’ in (T)
go
```

此示例使用 replicate 'UD' 子句在所有表上启用 **update** 和 **delete** 语句：

```
create database replication definition dbrepdef_UD
       with primary at ds2.pdb1
       replicate 'UD'
go
```

您可以在同一定义中多次使用多个子句指定表。不过，在每个定义中只能使用一次 **U**、**D**、**I** 和 **S**。

```
create database replication definition dbrepdef
       with primary at ds2.pdb1
       replicate tables in (tb1,tb2)
       replicate 'U' in (tb1)
       replicate 'I' in (tb1,tb2)
go
```

以下示例为 tb1 和 tb2 表应用 **update** 和 **delete** 语句：

```
alter database replication definition dbrepdef
      with primary at ds1.pdb1
      replicate 'UD' in (tb1,tb2)
go
```

表复制定义

要支持 SQL 语句复制，您必须在创建表复制定义时包括 **replicate SQLDML** 子句。

create replication definition 语法必须包括以下子句：

```
[replicate {SQLDML [ 'off' ] | 'options' }]
```

'options' 参数是以下这些语句的组合：

- **U** – update
- **D** – delete
- **I** – insert select

注意： 如果您的复制定义中包含 **[replicate {minimal | all} columns]** 子句，那么 **[replicate {minimal | all} columns]** 子句必须始终位于 **[replicate {SQLDML ['off'] | 'options' }]** 子句之前。

下面是针对某个表的 **create replication definition** 示例：

```
create replication definition repdef1
       with primary at ds3.pdb1
       with all tables named 'tb1'

           (id_col int,
           str_col char(40))

       primary key (id_col)
       replicate all columns
       replicate 'UD'
go
```

包含 **send standby** 子句的表复制定义可以指定 **replicate ‘I’** 语句。只能在在热备份或 MSA 环境中将 **insert select** 语句作为 SQL 复制语句进行复制。不包含 **send standby** 子句的表复制定义无法复制 **insert select** 语句。

热备份和 SQL 语句复制

了解如何为 SQL 语句复制配置热备份应用程序支持。

缺省情况下，热备份应用程序不会复制支持 SQL 语句复制的 DML 命令。要使用 SQL 复制，您可以：

- 使用 **replicate SQLDML** 和 **send standby** 子句创建表复制定义。
- 将 **WS_SQLDML_REPLICATION** 参数设置为 on。缺省值为 **UDIS**。
  不过，**WS_SQLDML_REPLICATION** 的优先级比 SQL 复制的表复制定义低。如果表的表复制定义包含 **send standby** 子句，该子句将确定是否复制 DML 语句，而无论 **WS_SQLDML_REPLICATION** 参数设置是什么。

SQL 语句复制的行计数验证

可以指定 Replication Server 如何响应在 SQL 语句复制期间可能发生的 SQLDML 行数错误。

当 SQL 语句复制后主数据库与复制数据库中更改的行数不相符时，便会出现 SQLDML 行计数错误。缺省错误操作为停止复制。

若要为 SQLDML 行数错误指定其它错误操作，请在主节点上为 Replication Server 错误类执行 **assign action** 命令：

```
assign action
   {ignore | warn | retry_log | log | retry_stop | stop_replication}
   for error_class
   to server_error1 [, server_error2]...
```

其中：

- *error_class* 是将操作指派给的错误类名称。您可以指定 Replication Server 错误类，例如缺省的 **rs_repserver_error_class** 错误类。
- *server_error* 是错误号。您可以为 Replication Server 指定错误号。

例如，要指定在 Replication Server 遇到错误号 5186 执行的 **warn** 错误操作，请输入：

```
assign action warn for rs_repserver_error_class to 5186
```

如果出现行计数错误，下面是生成的错误消息示例：

```
DSI_SQLDML_ROW_COUNT_INVALID 5186
Row count mismatch for SQLDML command executed on
'mydataserver.mydatabase'.
The command impacted 1000 rows but it should impact 1500 rows.
```

另请参见

- 数据服务器错误处理 （第 267 页）

SQL 语句复制的错误操作

Replication Server 支持一些针对 SQL 语句复制错误的错误操作。

表 23. SQL 语句复制的错误操作

server_error	错误消息	缺省错误操作	说明
5186	`Row count mismatch for the command executed on 'dataserver.database'. The command impacted x rows but it should impact y rows.`	**stop_replication**	SQL 语句复制影响的行数与预期的影响行数不相同时发生的行数验证错误。
5193	`You cannot enable autocorrection if SQL Statement Replication is enabled. Either enable SQL Statement Replication only or disable SQL StatementReplication before you enable autocorrection.`	**stop_replication**	在启用 SQL 语句复制的情况下，无法启用自动更正。请仅启用 SQL 语句复制，或者在启用自动更正之前禁用 SQL 语句复制

SQL 语句复制作用域

了解 SQL 语句复制如何应用于批处理、触发器和存储过程中的 DML 语句。

批处理

在将 SQL 语句复制应用于任何批量执行的 DML 语句时，应满足一些要求。

- 配置必须允许使用 SQL 语句复制。
- DML 语句不符合使用 SQL 语句复制的任何条件或例外情况。

在下面的示例中，在使用 **delete** 和 **insert** 执行批处理语句时，仅第一个语句使用 SQL 语句复制。`table2` 使用传统复制，因为 `table2` 未配置为使用 SQL 语句复制：

```
create table table1 (c int, d char(5))
go
create table table2 (c int, d char(5))
go
insert table1 values (1, 'ABCDE')
go 100
sp_setreptable table1, true
go
sp_setreptable table2, true
go
sp_setrepdefmode table1, 'UDI', 'on'
go
delete table1 where c=1
```

```
insert table2 select * from table1
go
```

另请参见

- 使用 SQL 语句复制的例外情况 （第 190 页）

存储过程

存储过程的复制状态决定了是否将其中的 DML 语句作为语句进行复制。

- 如果存储过程未标记为要复制，存储过程中的 DML 语句将作为语句进行复制，但前提是：
  - 配置允许使用 SQL 语句复制。
  - DML 语句不符合使用 SQL 语句复制的任何条件或例外情况。
- 如果存储过程标记为要复制，则仅复制对它的调用，而不复制组成存储过程的各个语句。

另请参见

- 使用 SQL 语句复制的例外情况 （第 190 页）

触发器

在 Adaptive Server 针对触发器中的 DML 语句使用 SQL 语句复制时，应满足一些要求。

- 配置允许使用 SQL 语句复制。
- DML 语句不符合使用 SQL 语句复制的任何条件或例外情况。

在下面的示例中，在 `table1` 上执行 **delete** 语句时，将使用传统复制方法复制该语句。通过触发器在 `table2` 上执行的 **delete** 将使用 SQL 语句复制进行复制，原因是为该表配置了 SQL 语句复制并且 **delete** 符合以语句形式进行复制的条件：

```
create table table1 (c int, d char(5))
go
create table table2 (c int, d char(5))
go
sp_setreptable table1, true
go
sp_setreptable table2, true
go
insert table1 values (1,'one')
go
insert table2 values (2,'two')
go 100
sp_setrepdefmode table2, 'udi', 'on'
go
create trigger del_table1 on table1
for delete
as
begin
delete table2
```

```
end
go
delete table1 where c=1
go
```

另请参见

- 使用 SQL 语句复制的例外情况 （第 190 页）

重新编译存储过程和触发器

如果在两次连续执行存储过程或触发器之间 SQL 复制设置从“off”更改为“on”，则会自动重新编译存储过程和触发器。

编译时的 SQL 语句复制设置	运行时的 SQL 语句复制设置	是否自动重新编译存储过程/触发器？
Off	ON	是
ON	Off	否

跨数据库的事务

单个事务可能会影响不同数据库中的表。位于不同数据库中的表的修改将记录在保存这些表的数据库中。

为数据库配置的 Sybase Replication Agent 将发送其事务日志中存储的 Replication Server 信息。

在此示例中，db1 和 db2 是具有配置的 Sybase Replication Agent 的复制型数据库。db1 数据库配置为使用 SQL 语句复制：

```
use db2
go
begin tran
go
delete t1 where c between 1 and 10000000
delete db1..t1 where c between 1 and 1000000
commit tran
go
```

第二个 **delete**（在 db1 数据库上）使用 SQL 语句复制，而第一个 **delete**（在 db2 数据库上）使用传统复制。在 db1 上运行的 Sybase Replication Agent 将复制该语句。

Replication Server 不保证不同数据库之间的事务完整性。例如，如果第一个 **delete** 的 DSI 挂起，而第二个 **delete** 的 DSI 处于活动状态，则第二个 **delete** 在第一个 **delete** 之前进行复制。

SQL 语句复制解决的问题

在某些情况下，无法使用传统复制方法复制数据。SQL 语句复制提供了一种在这些情况下成功复制数据的方法。

在热备份配置中复制 select into 操作

select into 根据 **select** 列表中指定的列以及在 **where** 子句中选择的行来创建新表。将对此操作进行最少的记录以便于恢复，并且无法使用传统复制方法进行复制。

可以在热备份配置中使用 SQL 语句复制方法复制 **select into**。要在数据库级配置 SQL 语句复制，请使用：

```
sp_setrepdbmode pdb, 'S', 'on'
go
```

当此选项在数据库级处于活动状态后，将使用 SQL 语句复制方法复制 `pdb` 数据库中的所有 **select into** 操作。查看使用 SQL 语句复制的例外情况，以确认是否可以使用 SQL 语句复制方法复制查询。如果只需要复制 **select into** 的子集，请改用 **set repmode**。

另请参见

- 使用 SQL 语句复制的例外情况 （第 190 页）

复制主键上的延迟更新

传统复制不支持具有唯一列索引的表上的更新，Replication Server 将报告错误。

例如，`t` 表在 `c` 上具有唯一索引，并且具有以下值：1、2、3、4 和 5。将在表中应用单个 **update** 语句：

```
update t set c = c+1
```

在使用传统复制时，此语句将变为：

```
update t set c = 2 where c = 1
update t set c = 3 where c = 2
update t set c = 4 where c = 3
update t set c = 5 where c = 4
update t set c = 6 where c = 5
```

第一个 update 尝试将 c=2 值插入到表中；不过，该值在表中已存在。Replication Server 显示错误 2601，即，尝试插入重复的键。

您可以使用 SQL 语句复制解决该问题。如果表具有唯一索引，并且为 **update** 语句配置了 SQL 语句复制，Adaptive Server 将使用 SQL 语句复制方法复制更新。

使用 SQL 语句复制的例外情况

使用 SQL 语句复制存在一些限制。

在下列情况下，不支持 SQL 语句复制：

- 复制数据库的表模式与主数据库不同。
- Replication Server 必须执行数据或模式转换。
- 预订或项目包含 **where** 子句。
- 更新包含一个或多个 `text` 或 `image` 列。

- 已自定义函数字符串 `rs_delete`、`rs_insert` 和 `rs_update`。
- DML 语句与此处列出的一个或多个条件相匹配。在这些情况下，将使用传统复制：
  - 语句引用视图、临时表或其它数据库中的表。

    ```
    insert tbl select * from #tmp_info
    where column = 'remove'
    ```
  - 用户在将 **set rowcount** 选项设置为大于零的值的情况下执行语句。

    ```
    set rowcount 1
    update customers
    set information = 'reviewed'
    where information = 'pending'
    ```
  - 语句在 **select** 或 **select into** 语句、Java 函数或 SQL 用户定义的函数 (UDF) 中使用 **top n** 子句：

    ```
    delete top 5
    from customers
    where information = 'obsolete'
    ```
  - 基表包括加密列，并且语句在 **set** 或 **where** 子句中引用这些列之一。
  - 语句引用系统目录或虚设表，例如，“deleted”或“inserted”。在此示例中，触发器执行的 **delete** 不使用 SQL 语句复制，因为它使用虚设表 `deleted`：

    ```
    create trigger customers_trg on customers for delete as
    delete customers_hist
    from customers_hist, deleted
    where deleted.custID = customers_hist.custID
    go
    delete customers where  state = 'MA'
    go
    ```
  - 语句是生成新的 `identity` 或 `timestamp` 值的 **insert** 语句。
  - 语句是更改 `timestamp` 或 `identity` 值的 **update** 语句。
  - 语句是为局部变量分配值的 **update** 语句。例如：

    ```
    update t set @a = @a + 2, c = @a where c > 1
    ```
  - 语句引用实现计算列。
  - 语句是影响具有加密列的复制表的 **select into** 语句。
  - 语句是使用 **union** 子句的 **insert select** 或 **select into**：

    ```
    select c1, c2 from tbl2
    union
    select cc1, cc2 from tbl3
    ```
  - 语句是具有 `text/image` 列的表上的 **update**、**insert select** 或 **select into**。
  - 语句是使用内置函数的查询：

    如果内置函数可以分析为一个常量值，则会将查询作为 SQL 语句进行复制。例如：

    ```
    update tbl set value = convert(int, "15")
    ```

    不过，不会使用 SQL 语句复制方法复制以下查询：

    ```
    update tbl set value = convert(int, column5)
    ```

在热备份拓扑中，可以使用 SQL 语句复制方法复制包含以下内置函数的查询，即使内置函数无法分析为常量值：

abs	cot	ltrim	sqrt
acos	datalength	patindex	str
ascii	degrees	power	strtobin
asin	exp	replicate	stuff
atan	floor	reverse	substring
atn2	hextoint	right	tan
bintostr	inttohex	round	to_unichar
ceiling	len	rtrim	upper
char	log	sign	
convert	log10	soundex	
cos	lower	space	

SQL 语句复制不支持自动更正

SQL 语句复制无法执行自动更正。如果数据服务器接口 (DSI) 遇到 SQL 语句复制的 DML 命令且自动更正为 on，缺省情况下，DSI 挂起并停止复制。

可以将 **assign action** 命令与错误号 5193 一起使用以指定 Replication Server 如何处理该错误。

直到表级预订通过验证，Replication Server 才会复制 SQLDML。

RSSD 系统表修改

可以对 Replication Server 系统数据库 (RSSD) 中的系统表进行一些更改以支持 SQL 语句复制。

RSSD 中的这些系统表支持 SQL 语句复制：

- `rs_dbreps – status` 列包含 4 组新的 2 位集，每一组对应于一个 DML 过滤器。集的第一位表示其是否为空过滤器，第二位表示其是否为否定语句集。
- `rs_dbsubsets – type` 列包含四种新类型：**U**、**L**、**I** 和 **S**，分别对应于 DML **UDIS** 过滤器。在这种情况下，将 **L** 而不是 **D** 用于删除。
- `rs_objects – attributes` 列包含五个新位：每个 **U**、**D**、**I** 或 **S** 操作各一个位，还有一个位指示表复制定义的列数是否比传入数据行少。

  系统函数复制定义 **rs_sqldml** 也支持 SQL 语句复制。

用于 SQL 语句复制的 Adaptive Server 监控表

可以在 SQL 语句复制期间使用 Adaptive Server 监控表来提供 Adaptive Server 状态的统计快照。可以通过这些表分析 Adaptive Server 和 Adaptive Server 性能。

表	说明
monSQLRepActivity	为使用 SQL 语句复制功能复制的 DML 语句上的所有打开的对象提供统计信息。
monSQLRepMisses	为未使用 SQL 语句复制功能复制的操作提供统计信息。threshold、querylimitation 和 configuration 列指示这些因素之一禁止对象使用 SQL 语句复制的次数。

请参见《Adaptive Server Enterprise 性能和调优系列：监控表》的“监控表简介”中的“Adaptive Server 中的监控表”。

产品版本和混合版本要求

SQL 语句复制要求 Adaptive Server 15.0.3 版和更高版本、主 Replication Server 和复制 Replication Server 15.2 版和更高版本以及路由 15.2 版和更高版本。

降级和 SQL 语句复制

了解在使用 SQL 语句复制并希望将 Adaptive Server 降级到早于 15.0.2 ESD #3 的版本或将 Replication Server 降级到早于 15.2 的版本时的降级过程。

Adaptive Server 降级

您可以将 Adaptive Server 降级到较早的版本，同时仍在日志中保存与 SQL 语句复制有关的事务记录。

如果要降级到低于 15.0.2 ESD #3 的版本，Sybase 建议您使用标准文档过程来降级使用复制型数据库的 Adaptive Server。此过程包括清除事务日志。请参见《Adaptive Server Enterprise 15.0.3 安装指南》。

Adaptive Server 15.0.3 为 Sybase Replication Agent 15.0.2 ESD #3 和更高版本提供以下降级支持：

- 即使日志包含有关 SQL 语句复制的信息，Sybase Replication Agent 也会继续复制数据。
- 当 Sybase Replication Agent 读取包含 SQL 语句复制的事务时，它发送该语句的原子修改并忽略与 SQL 语句复制有关的信息。

Replication Server 降级

您可以将 Replication Server 降级到早于 15.2 的版本。

Sybase Replication Agent 根据在建立连接时协商的日志传送语言 (LTL) 版本控制发送到 Replication Server 的信息数量和类型。

对于早于 15.2 的 Replication Server，Sybase Replication Agent 不发送 SQL 语句复制信息并使用标准复制方法进行复制。

适用于增强的 Replication Server 性能的动态 SQL

Replication Server 中的动态 SQL 通过允许 Replication Server 数据服务器接口 (DSI) 准备目标用户数据库上的动态 SQL 语句并重复执行这些语句来增强复制性能。

不是将 SQL 语言命令发送到目标数据库，而是仅在每次执行时发送实际值，因此消除了检查 SQL 语句语法和构建优化查询计划所产生的开销。此外，DSI 还通过以下方法优化动态 SQL 语句：仅在动态 SQL 命令失败时生成语言命令，并且仅在首次使用准备的语句时生成该语句。

如果启用了动态 SQL，在下列情况下，将在用户数据库连接中使用动态 SQL 而非语言命令：

- 命令为 **insert**、**update** 或 **delete**。
- 命令中没有 `text`、`image`、`java` 或 `opaque` 列。
- 在 **update** 或 **delete** 命令的 **where** 子句中没有 NULL 值。
- 命令中包含的参数没有超过 255 个：
  - **insert** 命令中可以包含不超过 255 的列。
  - **update** 命令在组合的 **set** 子句和 **where** 子句中可以包含不超过 255 的列。
  - **delete** 命令在 **where** 子句中可以包含不超过 255 的列。
- 该命令不使用用户定义的函数字符串。

动态 SQL 配置参数

可以使用动态 SQL 配置参数启用和调优动态 SQL。

- **dynamic_sql** – 为复制连接打开或关闭动态 SQL。其它动态 SQL 配置参数仅在设置为“on”时生效。
- **dynamic_sql_cache_size** – 告知 Replication Server 有多少数据库对象可以使用连接的动态 SQL。此参数限制数据服务器上的资源需求。
- **dynamic_sql_cache_management** – 管理连接的动态 SQL 高速缓存。一旦动态 SQL 语句到达某个连接的 **dynamic_sql_cache_size**，它就会停止对新动态 SQL 语句的分配（如果值为 **fixed**），或保留最近使用的语句并释放其余语句以分配新语句（如果值为 **mru**）。

设置配置参数以使用动态 SQL

可以在服务器或数据库连接级启用或配置动态 SQL。

缺省情况下，在服务器和连接级别上动态 SQL 处于关闭状态。

在服务器级设置动态 SQL 配置参数，以便为将来创建或启动的连接提供缺省值。要在服务器级配置动态 SQL，请输入：

```
configure replication server
set { dynamic_sql |
      dynamic_sql_cache_size |
      dynamic_sql_cache_management }
to value
```

要在连接级配置动态 SQL，请输入：

```
alter connection to server.db
set { dynamic_sql |
      dynamic_sql_cache_size |
      dynamic_sql_cache_management }
to value
```

表级动态 SQL 控制

可以使用 **create replication definition** 和 **alter replication definition** 控制通过复制定义在每个表上应用动态 SQL。

请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**create replication definition**”和“**alter replication definition**”。

您可以使用以下命令在表级为特定复制数据库更改动态 SQL 执行：

```
set dynamic_sql {on | off}
for replication definition with replicate at
data_server.database
```

在复制定义级别上，缺省为使用动态 SQL。只有在将表从动态 SQL 中排除时，才应更改缺省值。若要检查动态 SQL 用法，请打开 **stats_sampling**，运行 **admin stats, dsi** 命令，然后查看 DSIEDsqlPrepared、DSIEDsqlExecuted 和其它动态 SQL 计数器。

可以使用 **rs_helprep**、**rs_helpsub** 和 **rs_helppubsub** 显示每个复制定义的动态 SQL 设置。

请参见《Replication Server 参考手册》中的“RSSD 存储过程”。

replicate minimal columns 子句和动态 SQL

当复制定义包含 replicate minimal columns 或将连接的 **replicate_minimal_columns** 设置为 on 时，复制处理将使用动态 SQL。

可以针对物理连接和热备份环境使用 **replicate_minimal_columns**。DSI 可以使用该参数确定在没有复制定义或复制定义不包含 **replicate minimal columns** 子句时是否使用最少的列数。

缺省情况下，将所有连接的 **replicate_minimal_columns** 设置为 on。连接的 **replicate_minimal_columns** 设置覆盖通过 **replicate all columns** 子句设置的复制定义。

对于自定义函数字符串，在将连接的 **replicate_minimal_columns** 设置为 on 时，当前复制环境的行为可能会发生变化。如果应用程序依靠要发送到复制数据库的命令进行触发器处理，在没有对行中的任何列进行更改时，**replicate_minimal_columns** 的缺省

设置“on”不会发送命令。要恢复最初的行为，请将连接的 **replicate_minimal_columns** 设置为 off。

例如，若要为到 SYDNEY_DS 数据服务器中的 pubs2 数据库的连接启用 **replicate_minimal_columns**，请输入：

```
alter connection to SYDNEY_DS.pubs2
set replicate_minimal_columns to ‘on’
```

如果预计即使没有更改行中的任何列的值也会引发触发器，**replicate_minimal_columns** 可能会影响触发器处理。

可以使用 **admin config** 显示 **replicate_minimal_columns** 设置。

注意： 在将 **dsi_compile_enable** 设置为“on”时，Replication Server 会忽略 **replicate_minimal_columns** 设置。

动态 SQL 的限制

在使用动态 SQL 时，应注意一些限制。

- 如果使用内部复制定义将某个表复制到备用连接或 MSA 连接，并为连接启用动态 SQL，则该表的任何新的复制定义都应定义与主数据库中的列顺序一致的列顺序。否则，可能会导致现已准备好的语句无效，并可能要求重新启动备用或 MSA 连接。
- Replication Server 在动态 SQL 命令中将用户定义的数据类型转换为 Open Client/Server™ (OCS) 数据类型。

  如果数据超出 Sybase 范围导致动态 SQL 失败，则 DSI 会记录一条错误消息并使用语言命令重新发送动态 SQL。DSI 仅在语言命令也失败时才会关闭。

  如果这种情况频繁发生，请通过表复制定义或使用 **set dynamic_sql off** 命令禁用动态 SQL。

禁用动态 SQL

可以使用几个命令来禁用动态 SQL。

- **alter connection... set dynamic_sql off** – 为此连接中的所有命令关闭动态 SQL。
- **create/alter replication definition...without dynamic_sql** – 为所有使用此复制定义的命令关闭动态 SQL。
- **set dynamic_sql off for replication definition with replicate at...** – 为在此复制连接处使用此复制定义的所有命令关闭动态 SQL。

高级服务选件

Replication Server 包括高级服务选件，其中包含复制性能的增强功能。

要在高级服务选件中激活任何增强功能，您必须具有 REP_HVAR_ASE 许可证文件。请参见《Replication Server 安装指南》的“规划安装”中的“获取许可证”。

另请参见

- 向 Adaptive Server 进行高容量自适应复制 （第 197 页）
- 增强的重试机制 （第 204 页）
- 提高的 DSI 效率 （第 210 页）
- 提高的 RepAgent 执行程序线程效率 （第 211 页）
- 提高的分配器线程读取效率 （第 212 页）
- 增强的内存分配 （第 212 页）
- 增加队列块大小 （第 212 页）

向 Adaptive Server 进行高容量自适应复制

Replication Server 包括高容量自适应复制 (HVAR)，在复制到具有相同数据库模式的数据库时，它提供比当前连续复制模式更好的性能。

在连续复制模式下，Replication Server 根据主数据库中的日志顺序将每个记录的更改发送到复制数据库。通过使用以下功能，HVAR 可以获得更好的性能：

- 编译 – 按每个表以及每个 **insert**、**update** 和 **delete** 操作重新安排复制数据，并将操作编译为净行操作。
- 批量应用 – 使用净结果的最高效批量接口来批量应用编译操作的净结果。Replication Server 使用内存净更改数据库来存储更改，然后将其应用于复制数据库。

并非发送每个记录的操作，编译删除操作组中的中间 **insert**、**update** 或 **delete** 操作，只发送复制的事务的最终编译状态。根据事务配置文件，这通常意味着 Replication Server 向复制数据库发送少量命令进行处理。

HVAR 将尽可能多的可编译事务分组在一起，将组中的事务编译为净更改，然后使用复制数据库中的批量接口将净更改应用于复制数据库。

在 Replication Server 编译大量事务并将其组合到组中时，批量操作处理得到了改进；因此，复制吞吐量和性能也得到了改进。您可以调整组大小来控制为批量应用分组在一起的数据量。

HVAR 尤其适用于创建联机事务处理 (OLTP) 归档和报告系统，其中复制数据库与主数据库具有相同的模式。

数据库和平台支持

HVAR 支持复制到 Adaptive Server 12.5 及更高版本中，您可以使用 64 位硬件平台实现最佳性能。

请参见《Replication Server 15.5 新增功能指南》的“Replication Server 15.5 版的新增功能”的“操作系统和平台支持”中的“64 位支持”。

HVAR 编译和批量应用

在编译过程中，HVAR 重新整理要复制的数据，方法是基于每个表以及每个 **insert**、**update** 和 **delete** 操作将数据集群在一起，然后将操作编译为净行操作。

HVAR 通过复制定义中定义的主键区分不同的数据行。如果没有复制定义，则将除 `text` 和 `image` 列以外的所有其它列视为主键。

对于普通复制环境中发现的操作组合，如果假定表和行具有相同的主键，HVAR 将遵循以下操作编译规则：

- **insert** 后跟 **delete** 不会导致任何操作。
- **delete** 后跟 **insert** 不会导致简化。
- **update** 后跟 **delete** 将导致 **delete**。
- **insert** 后跟 **update** 将导致 **insert**，即两项操作简化为一项最终操作，该操作包含被第二项操作中的所有差异覆盖的第一项操作的结果。
- **update** 后跟另一个 **update** 将导致 **update**，即两项操作简化为一项最终操作，该操作包含被第二项操作中的所有差异覆盖的第一项操作的结果。

其它操作组合将导致无效的编译状态。

示例 1

这是日志顺序的逐行更改示例。在此示例中，`T` 是先前由以下命令创建的一个表：

create table `T(k int , c int)`

```
1. insert T values (1, 10)
2. update T set c = 11 where k = 1
3. delete T where k = 1
4. insert T values (1, 12)
5. delete T where k =1
6. insert T values (1, 13)
```

通过使用 HVAR，可以将 1 中的 **insert** 和 2 中的 **update** 转换为 **insert** `T 值 (1, 11)`。转换后的 **insert** 和 3 中的 **delete** 相互取消，可以将其删除。可以删除 4 中的 **insert** 和 5 中的 **delete**。最终编译的 HVAR 操作为 6 中的最后一个 **insert**：

```
insert T values (1, 13)
```

示例 2

在另一个日志顺序的逐行更改示例中：

```
1. update T set c = 14 where k = 1
2. update T set c = 15 where k = 1
3. update T set c = 16 where k = 1
```

通过使用 HVAR，1 和 2 中的 **update** 可简化为 2 中的 **update**。2 和 3 中的 update 可简化为 3 中的单个 **update**，这是 k = 1 的净行更改。

Replication Server 使用内存净更改数据库中的 **insert**、**delete** 和 **update** 表来存储应用于复制数据库的净行更改。净行更改按复制表和操作类型（**insert**、**update** 或 **delete**）排序，然后为批量接口做准备。HVAR 直接将 **insert** 操作装载到复制表中。由于

Adaptive Server 不支持批量 **update** 和 **delete**，因此，HVAR 将 **update** 和 **delete** 操作装载到 HVAR 在复制数据库中创建的临时工作表中。然后，HVAR 对复制表执行 **join-update** 或 **join-delete** 操作以获取最终结果。工作表是动态创建和删除的。

在示例 2 中，编译导致 `update T set c = 16 where k = 1`:

1. HVAR 创建 `#rs_uT(k int, c int)` 工作表。
2. HVAR 使用以下语句在工作表中执行 **insert** 操作:

   ```
   insert into #rs_uT(k, c) location 'idemo.db' {select * from
   rs_uT}
   ```

3. HVAR 执行 **join-update**:

   ```
   update T set T.c=#rs_uT.c from T,#rs_uT where T.k=#rs_uT.k
   ```

在 HVAR 编译大量事务并将其组合到组中时，批量操作处理得到了改进；因此，复制吞吐量和性能也得到了改进。您可以通过使用配置参数调整 HVAR 大小来控制 HVAR 为批量应用分组在一起的数据量。

虽然 HVAR 应用行更改的顺序与更改的记录顺序不同，但没有数据丢失，原因是:

- 对于不同的数据行，应用行更改的顺序不会影响结果。
- 在同一行中，在编译后在 **insert** 之前应用 **delete** 可保持一致性。

净更改数据库

Replication Server 具有充当用于存储事务的净行更改（即编译的事务）的内存存储库的净更改数据库。

每个事务有一个净更改数据库实例。每个复制表最多可以在净更改数据库中具有三个跟踪表。您可以检查净更改数据库和该数据库中的表来监控 HVAR 复制和解决问题。

监控净更改数据库

可以使用 **sysadmin cdb** 命令监控净更改数据库和访问净更改数据库实例。

请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**sysadmin cdb**”。

HVAR 处理和限制

HVAR 只应用事务的净行更改同时保持原始提交顺序，并保证事务一致性，即使它跳过中间行更改也是如此。

这具有以下几种含义:

- 为避免 HVAR 消耗较高的内存量，Replicaton Server 通过连续复制模式处理并应用大事务，而不是通过 HVAR 进行处理。大事务的阈值取决于可使用 **dsi_sqt_max_cache_size** 设置的 SQT 高速缓存大小以及可使用 **dsi_compile_max_cmds** 和 **dsi_cdb_max_size** 控制的净更改数据库大小。
- **Insert** 触发器不触发，因为 HVAR 进程直接将净新行批量装载到表中。**Update** 和 **delete** 触发器在 Replication Server 将编译的净结果应用到复制数据库时继续触发。

但是，Replication Server 编译的行修改以及不再位于净结果中的行修改对于触发器不可见。触发器只能检测到最终行图像。

假设您使用 Replication Server，通过将用户与用户修改的表中的任意列关联的触发器逻辑，借助表模式中的 `last_update_user` 列，来审计用户更新。如果 userA 修改表中的 `colA` 和 `colC`，然后 userB 修改 `colB` 和 `colD`，则当触发器触发时，触发器逻辑只能检测到最后一个修改该表的用户，因此触发器逻辑将 userB 作为最后一个修改所有四列的用户进行关联。如果您定义包含类似逻辑的触发器，其中每个行修改都必须被检测到，则可能必须对该表禁用 HVAR 编译。

- HVAR 应用行更改的顺序与行更改的记录顺序不同。若要按日志顺序对复制表应用更改，请为该表禁用 HVAR 编译。
- 如果对复制表有参照约束，则必须在复制定义中指定约束。为避免约束错误，HVAR 根据复制定义装载表。
- 在启用 HVAR 时，Replication Server 不支持自定义函数字符串或任何并行 DSI 序列化方法，缺省的 **wait_for_commit** 方法除外。HVAR 将自定义函数字符串视为不可编译的命令。
- Replication Server 在遇到以下内容时恢复为日志顺序逐行连续复制。
  - 不可编译的命令 – 存储过程、SQL 语句、数据定义语言 (DDL) 事务、系统事务和 Replication Server 内部标记。
  - 不可编译的事务 – 包含不可编译的命令的事务。
  - 不可编译的表 – 禁用了 HVAR 的表、具有修改的函数字符串的表、具有参照约束关系的表和 HVAR 无法编译的表。
- 如果复制定义不包括 **replicate minimal columns** 子句，则 HVAR 自动将主键 **update** 更改为 **delete** 后跟 **insert**。主键更新为以下内容之一：
  - 影响在表的复制定义中定义的主键的更新，或者
  - 在不存在复制定义时影响 `text` 和 `image` 以外的列的更新。在这种情况下，Replication Server 假定所有列是主键的一部分，因为复制定义中没有具体的主键定义。
- 如果复制定义包括 **replicate minimal columns** 子句，并且 HVAR 编译的组包含可修改主键列的 **update** 命令，则 HVAR 在运行时针对组的其余部分自动将表指定为不可编译。应用于表的 **update** 操作不可编译，因为 HVAR 无法将 **update** 转换为由 **delete** 和 **insert** 组成的一对操作。在 HVAR 处理的事务组中，HVAR 可以成功将遇到不可编译的主键 **update** 操作之前处理的所有操作编译为净更改数据库。不过，在事务组中，HVAR 将初始不可编译主键 **update** 及其后面的所有操作标记为不可编译。表的不可编译状态是短暂的，仅持续到 HVAR 处理的相同事务组的持续时间结束。
- HVAR 忽略可停止事务分组的参数，如 **dsi_partition_rule**。
- 如果在 HVAR 处理期间出现错误，则 Replication Server 将通过逐渐缩小事务组来重试编译，直到识别未通过编译的事务，然后使用连续复制应用该事务。
- 要实现性能优点，请保持主数据库和复制数据库同步以避免 Replication Server 在出错时进行其它处理的开销。可以将 **dsi_command_convert** 设置为 **i2di,u2di** 以便

同步数据，但这也会造成处理开销。如果数据库同步，请将 **dsi_command_convert** 重置为 **none**。

- HVAR 执行行计数验证以确保复制完整性。行计数验证基于编译。预期的行计数是编译后剩余的行数。
- 当复制定义中存在 `identity` 数据类型的列时，Replication Server 在复制数据库中执行以下命令：
  - identity 列 insert 之前的 **set identity_insert**_*table_name* **on** 和 identity 列 insert 之后的 **set identity_insert**_*table_name* **off**。
  - identity 列 update 之前的 **set identity_update**_*table_name* **on** 和 identity 列 update 之前的 **set identity_update**_*table_name* **off**。

另请参见

- 具有参照约束的表 （第 207 页）

启用 HVAR

可以使用 **dsi_compile_enable** 为到复制数据库的连接启用 HVAR。

如果将 **dsi_compile_enable** 设置为 off，则 Replication Server 将使用连续日志顺序的逐行复制模式。例如，如果复制净行更改引发问题，则为受影响的表将 **dsi_compile_enable** 设置为 off，例如表上的触发器要求对表执行的所有操作按日志顺序复制，因而不允许编译。

注意： 如果将 **dsi_compile_enable** 设置为 on，Replication Server 将禁用 **dsi_cmd_prefetch** 和 **dsi_num_large_xact_threads**。

若要在数据库级启用和配置 HVAR 以只影响指定的数据库，请输入：

```
alter connection to data_server.database
set dsi_compile_enable to 'on'
go
```

也可以在服务器或表级启用和配置 HVAR。

- 服务器级 – 影响与 Replication Server 的所有数据库连接：

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

- 表级 – 只影响您指定的复制表。如果您同时在表级和数据库级指定参数，则表级参数优先于数据库级参数。如果您未指定表级参数，则参数的设置在数据库级应用。要为表设置参数，请使用 **alter connection** 和 **for replicate table named** 子句，例如：

  ```
  alter connection to data_server.database
  for replicate table named dbo.table_name
  set dsi_compile_enable to 'on'
  ```

  使用 **for replicate table name** 子句更改表级连接配置。配置更改用于从您指定的表的所有预订和所有复制定义复制数据。

注意： 对于表级配置，您只能使用 **alter connection**，因为 Replication Server 不支持带有 **for** 子句的 **create connection**。

在执行 **dsi_compile_enable** 后，挂起并重新开始与复制数据库的连接。

HVAR 配置参数

Replication Server 自动为多个参数设置 Sybase 建议的缺省值。可以更改这些参数的值以调优复制性能。

您必须为要更改的每个参数执行单独的 **alter connection** 命令。在输入 **alter connection** 后，不要输入多个参数。

HVAR 自动为 **dsi_cdb_max_size**、**dsi_compile_max_cmds**、**dsi_bulk_threshold**、**dsi_command_convert** 和 **dsi_compile_retry_threshold** 设置 Sybase 建议的缺省值。不过，可以在您的复制环境中指定自己的值以调优性能：

有关这些参数的完整说明，请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**alter connection**”。

dsi_bulk_threshold

dsi_bulk_threshold 指定在命令类型的表上进行编译后的净行更改命令数，在达到该数量时，将触发 Replication Server 对相同命令类型的该表使用批量拷入。

缺省值为 20 个净行更改命令。

示例：

```
alter connection to SYDNEY_DS.pubs2
set dsi_bulk_threshold to ‘15’
go
```

dsi_cdb_max_size

dsi_cdb_max_size 指定在事务不超过 DSI SQT 高速缓存或事务中的命令数不超过 **dsi_compile_max_cmds** 时 HVAR 可以编译的事务的最大大小 (MB)。

当 HVAR 编译的当前组中的事务大小达到 **dsi_compile_max_cmds** 时，HVAR 将启动新的组。如果没有其它要读取的数据，那么即使组没有达到 **dsi_cdb_max_size** 设置中的最大大小，HVAR 也会将当前事务集划分到当前组中。

缺省值为 1024MB。

示例：

```
alter connection to SYDNEY_DS.pubs2
set dsi_cdb_max_size to ‘2048’
go
```

dsi_compile_max_cmds

dsi_compile_max_cmds 指定在事务不超过 DSI SQT 高速缓存或事务大小不超过 **dsi_cdb_max_size** 时 HVAR 可以编译的事务的最大大小（命令数）。Replication Server 通过连续复制模式复制不可编译的事务。

当 HVAR 编译的当前组中的命令数达到 **dsi_compile_max_cmds** 时，HVAR 将启动新的组。如果没有其它要读取的数据，那么即使组没有达到 **dsi_compile_max_cmds** 中设置的最多命令数，HVAR 也会将当前事务集划分到当前组中。

缺省值为 10,000 个命令。

示例：

```
alter connection to SYDNEY_DS.pubs2
set dsi_compile_max_cmds to ‘50000’
go
```

dsi_compile_retry_threshold

dsi_compile_retry_threshold 指定组中的命令数阈值。如果包含失败事务的组中的命令数小于 **dsi_compile_retry_threshold** 值，则 Replication Server 不会重新尝试在 HVAR 模式下处理组，从而节约了处理时间并提高了性能。Replication Server 将为组切换到连续复制模式。连续复制模式根据主数据库日志顺序将每个记录的更改发送给复制数据库。

缺省值为 100 个命令。

示例：

```
alter connection to SYDNEY_DS.pubs2
set dsi_compile_retry_threshold to '200'
go
```

dsi_command_convert

dsi_command_convert – 指定如何转换复制命令。以下操作的组合指定转换类型：

- **d** – delete
- **i** – insert
- **u** – update
- **t** - truncate
- **none** – 无操作

dsi_command_convert 的操作组合包括 **i2none**、**u2none**、**d2none**、**i2di**、**t2none** 和 **u2di**。转换前的操作在“2”之前，转换后的操作在“2”之后。例如：

- **d2none** – 不复制 **delete** 命令。在使用此选项时，如果不希望复制 **delete** 操作，则无需自定义 **rs_delete** 函数字符串。
- **i2di,u2di** – 将 **insert** 和 **update** 都转换为 **delete** 后跟 **insert**，相当于自动更正。如果通过将 **dsi_row_count_validation** 设置为 off 来禁用行计数验证，则 Sybase 建议您将 **dsi_command_convert** 设置为 **i2di,u2di** 以避免重复键错误并允许在复制期间自动同步数据库。
- **t2none** – 不复制 **truncate table**。

dsi_command_convert 的缺省值为 **none**，这表示不进行命令转换。

示例：

```
alter connection to SYDNEY_DS.pubs2
set dsi_command_convert to 'i2di,u2di'
go
```

增强的重试机制

增强的重试机制通过减少 Replication Server 重试编译和批量应用的次数来提高复制性能。

HVAR 尝试将尽可能多的可编译事务分组在一起，将组中的事务编译为净更改，然后使用复制数据库中的批量接口将净更改应用于复制数据库。当 HVAR 处理产生的复制事务失败时，HVAR 将调用重试机制。如果组中的事务失败，则 HVAR 将该组拆分为两个大小相同的较小组，然后对每个组重试编译和批量应用。重试机制可识别失败的事务，允许 Replication Server 执行错误操作映射并可在 DSI 关闭时应用失败的事务之前的所有事务。

HVAR 中的净更改数据库充当用于存储事务的净行更改（即编译的事务）的内存存储库。净更改数据库的内容是 HVAR 没有按日志顺序应用的不同主事务中的命令集合。因此，不使用重试机制，便无法识别失败的事务。只要组中的事务失败，重试机制便拆分组并连续重试编译和批量应用。此连续重试过程会降低性能。

当 HVAR 遇到包含失败的事务的组时，增强的重试机制将该组拆分为三个大小相同的组，以便该机制能够更高效地识别包含失败事务的组。

另外，还可以使用 **dsi_compile_retry_threshold** 参数为组中的命令数指定阈值。如果包含失败事务的组中的命令数小于 **dsi_compile_retry_threshold** 值，则 Replication Server 不会重新尝试在 HVAR 模式下处理组，从而节约了处理时间并提高了性能。Replication Server 将为组切换到连续复制模式。连续复制模式根据主数据库日志顺序将每个记录的更改发送给复制数据库。

内存消耗量控制

若要减少 HVAR 中的内存消耗量，请控制可编译的组大小。

内存消耗量是指 Replication Server 数据结构（如净更改数据库）及其存储的数据。净更改数据库是内存数据结构。在 Replication Server 编译应用于具有大量列的表或具有较大 `text` 和 `image` 数据类型值的表的命令时，净更改数据库内存消耗量可能会显著增加。例如，与编译具有 10 个列的表中的 1,000,000 行相比，编译具有 100 个列的表中的同样数量的行消耗的内存量可能大约多出 10 倍。如果可用于其它进程和模块的内存不足，复制性能将会受到影响。

如果事务大于 DSI SQT 高速缓存大小，则 Replication Server 将其标记为不可编译。如果事务可以放入 DSI SQT 高速缓存中，则 Replication Server 针对 **dsi_cdb_max_size** 和 **dsi_compile_max_cmds** 值检查该事务的大小。如果 Replication Server 估计事务所需的净更改数据库大小大于 **dsi_cdb_max_size**，或者事务包含的命令比 **dsi_compile_max_cmds** 多，则 Replication Server 将该事务标记为不可编译。Replication Server 在连续复制模式下应用这种不可编译的大事务。使用连续复制模式可避免生成单个大型净更改数据库以容纳大事务并降低内存消耗量。

Replication Server 尝试将尽可能多的可编译事务划分到一个可编译组中。Replication Server 还将 **dsi_cdb_max_size** 和 **dsi_compile_max_cmds** 作为可编译组的大小阈值。当组达到 **dsi_cdb_max_size** 或 **dsi_compile_max_cmds** 中设置的大小时，Replication Server 停止将事务编译到该组中，并将每个可编译的组作为单个事务应用于复制数据库。

HVAR 的 SQT 内存消耗量控制

在 HVAR 中的事务分析期间，控制 DSI SQT 高速缓存中的解包命令消耗的最大内存量。

SQT 线程监控 HVAR 事务分析进程解包的命令以及 DSI SQT 高速缓存引用的命令所消耗的内存量。

当 Replication Server 使用 HVAR 进行复制时，DSI 线程所消耗的最大内存量是 **dsi_sqt_max_cache_size**、**sqt_max_prs_size** 和 **dsi_cdb_max_size** 的总和。将 **dsi_sqt_max_cache_size**、**sqt_max_prs_size** 和 **dsi_cdb_max_size** 设置为较小的值将会减少内存消耗量，但会降低复制性能。调优复制环境以实现最佳的内存消耗量和性能。请参见《Replication Server 参考手册》中的“Replication Server 命令”以配置这些参数。

净更改数据库大小估计和事务分析

即使事务大于 DSI SQT 高速缓存大小，Replication Server 也不会将其标记为不可编译。

由于没有 DSI SQT 高速缓存大小约束，Replication Server 提高了估计净更改数据库大小的能力，避免了需要切换到连续复制模式并提高了事务分析过程的效率。

只有在以下情况时，Replication Server 才会将事务标记为不可编译：

- 事务中的命令数超过 **dsi_compile_max_cmds**，或者
- 估计的事务净更改数据库大小超过 **dsi_cdb_max_size**

为进一步提高性能，Replication Server 将估计净更改数据库大小的次数减少为每 100 个命令一次。

HVAR 的完全增量编译

完全增量编译通过在处理包含许多命令的大型可编译事务期间减少内存消耗量，提高了高容量自适应复制 (HVAR) 的复制性能。在使用完全增量编译时，Replication Server 不需要恢复为连续复制模式，而是使用更有效的 HVAR 模式来编译和复制大事务。

完全增量编译可以编译包含混合 **insert**、**delete** 或 **update** 操作的大事务。Replication Server 使用完全增量编译为复制数据库应用较大可编译事务（使用多个内存净更改数据库实例）。完全增量编译将大事务分成一系列段。每个段由一组命令组成。

Replication Server 编译每个段，并创建专用净更改数据库以存储一个段。Replication Server 指示净更改数据库实例发送段并将其应用于复制数据库。然后，Replication Server 关闭净更改数据库实例并释放消耗的内存。Replication Server 为下一个事务段

创建另一个净更改数据库实例，并继续按顺序为所有段创建和关闭净更改数据库实例。

因此，完全增量编译并非消耗大量内存以使大型净更改数据库实例保存大事务，而是将内存需求量减少到单个较小的净更改数据库实例（仅含一个事务段）所消耗的内存。完全增量编译将内存需求量除以使用的净更改数据库实例数。例如，当完全增量编译应用一个具有 10 个净更改数据库实例的大事务时，内存需求量大约为没有使用完全增量编译时的内存需求量的十分之一。

在编译小事务期间，由于 Adaptive Server 不支持批量 **update** 和 **delete**，因此，HVAR 将 **update** 和 **delete** 操作装载到 HVAR 在复制数据库内创建的临时工作表中。然后，HVAR 对复制表执行 **join-update** 或 **join-delete** 操作以获取最终结果。HVAR 动态创建和删除工作表。不过，要支持大事务编译使用完全增量编译，HVAR 使用常规表（而不是临时对象）在复制数据服务器上的 **tempdb** 数据库中创建工作表。

缺省情况下，Replication Server 不会为 HVAR 启用完全增量编译。

数据库支持
您可以为到 Adaptive Server 数据库的高容量自适应复制启用完全增量编译。

为 HVAR 启用完全增量编译
将 **RSFEATURE_HQ_INCR_CMPL_ON** 跟踪标志设置为 on，以便为到 Adaptive Server 复制数据库的高容量自适应复制 (HVAR) 启用完全增量编译。

登录到 **isql** 并将 **RSFEATURE_HQ_INCR_CMPL_ON** 跟踪标志设置为 on，以便为 HVAR 启用完全增量编译：

```
trace {"on"|"off"},rsfeature,rsfeature_hq_incr_cmpl_on
```

（可选）您可以在 Replication Server 配置文件末尾输入此跟踪标志：

```
trace=rsfeature,rsfeature_hq_incr_cmpl_on
```

要停用该跟踪标志，请从配置文件中删除该行。

缺省情况下，**RSFEATURE_HQ_INCR_CMPL_ON** 设置为 off。可以在 **isql** 中使用 **trace** 命令覆盖当前 Replication Server 会话的跟踪标志设置，即使该设置与配置文件中的设置不同。不过，在 Replication Server 重新启动时，它将执行配置文件中的设置。

内存控制参数和 Replication Server 处理
复制模式和操作取决于为内存控制参数设置的值。

Replication Server 处理

1. Replication Server 从出站队列中读取事务，并估计净更改数据库大小。
2. 如果事务仅包含 **insert**、**update** 和 **delete** 命令，则 Replication Server 将事务标记为可编译。
3. 在以下情况下，Replication Server 将事务标记为不可编译：
   - 事务中的命令数超过 **dsi_compile_max_cmds**，

- 估计的事务净更改数据库大小超过 **dsi_cdb_max_size**，或者
- 事务大小大于 DSI SQT 高速缓存大小。

Replication Server 在连续日志顺序模式下处理不可编译的事务。

4. 在 Replication Server 检查事务是否可编译后，它将后续可编译事务汇集到可编译组中。不过，Replication Server 根据两个阈值停止增加可编译组的大小：
   - 如果 Replication Server 计算它处理的可编译事务组中的命令数加上新事务中的命令数超过 **dsi_compile_max_cmds** 阈值，Replication Server 将关闭并分派该组，而将新事务添加到新的空组中。否则，Replication Server 将新的可编译事务添加到该组中。
   - 如果估计将新事务汇集到新组而产生的下一个净更改数据库的大小超过 **dsi_cdb_max_size**，Replication Server 将关闭并分派该组，而将新事务添加到新的空组中。否则，Replication Server 将新的可编译事务添加到该组中。
5. 如果出站队列中没有其它可编译事务，Replication Server 将立即关闭并分派所处理的组。Replication Server 不会等待新事务进入出站队列。

将 dsi_cdb_max_size 设置为不同的值

这些示例说明 Replication Server 在两个表上应用一个具有 100,000 个更新的事务。表 1 具有 100 个列，需要大约 4GB 的内存，表 2 具有 10 个列，需要大约十分之一的内存，即 400MB。

dsi_cdb_max_size 值 (MB)	表名	对复制处理的影响
1024（缺省值）	表 1	Replication Server 在连续日志顺序复制模式下应用事务。
1024（缺省值）	表 2	前提条件：将 Replication Server 中的 **memory_limit** 设置为足够大的值以允许构建 400MB 净更改数据库。 Replication Server 使用 HVAR 应用事务。
4096	表 1	Replication Server 使用连续日志顺序复制模式应用事务。
4096	表 2	前提条件：将 Replication Server 中的 **memory_limit** 设置为足够大的值以允许构建 400MB 净更改数据库。 Replication Server 使用 HVAR 应用事务。

具有参照约束的表

您可以使用复制定义来指定具有参照约束（例如外键及其它检查约束）的表，以便 HVAR 知道这些表。

通常，引用表包含相同主数据库中的被引用表的参照约束。不过，HVAR 将参照约束支持扩展到多个主数据库中的被引用表。

您可以在复制定义中为每个主数据库指定引用表。不过，如果多个参照约束相互冲突，则 Replication Server 会随机选择一个。

另请参见

- HVAR 处理和限制 （第 199 页）

复制定义创建和改变

可以使用带 **references** 参数的 **create replication definition** 命令指定具有参照约束的表。

create replication definition

```
   ...
   (column_name [as replicate_column_name]
       ...
       [map to published_datatype]] [quoted]
       [references [table_owner.]table_name [(column_name)]] …)
   ....]
```

可以使用带 **references** 参数的 **alter replication definition** 命令添加或更改引用表。可以使用 **null** 选项删除引用。

alter replication definition

```
    .....
add column_name [as replicate_column_name]
[map to published_datatype] [quoted]
[references [table_owner.]table_name [(column_name)]
...
| alter columns with column_name references
{[table_owner.]table_name [(column_name)] | NULL}
[, column_name references {[table_owner.]table_name [(column_name)]
| NULL}
      ...
```

对于带有 **reference** 子句的 **alter replication definition** 和 **create replication definition**，Replication Server 会：

- 将 **reference** 子句视为列属性。每个列只能引用一个表。
- 不处理您在 **reference** 子句中的 *column_name* 参数中提供的列名。
- 不允许具有循环引用的参照约束。例如，原始被引用表不能具有对原始引用表的参照约束。

在复制处理过程中，HVAR 装载：

- 对您在复制定义中指定的引用表之前的被引用表执行的插入操作。
- 对您在复制定义中指定的表之后的被引用表执行的删除操作。

在一些情况下，对两个表执行的更新操作因冲突而失败。为防止 HVAR 重试复制处理并因而降低性能，您可以：

- 通过将 **dsi_command_convert** 设置为“u2di”（这将更新转换为删除和插入），停止复制更新。
- 关闭 **dsi_compile_enable** 以避免编译受影响的表。

HVAR 无法编译具有自定义函数字符串的表和具有对它无法编译的现有表的参照约束的表。通过标记出这些表，HVAR 避免了由于参照约束错误引发的事务重试，从而优化了复制处理。

显示 HVAR 信息

您可以显示有关配置参数属性和表引用的信息。

显示配置参数属性

可以使用 **admin config** 查看有关数据库级和表级配置参数的信息，如示例中所示。

- 数据库级：
  - 若要显示到 NY_DS 数据服务器上的 `nydb1` 数据库 (`NY_DS.nydb1`) 的连接的所有数据库级配置参数，请输入：

    ```
    admin config, “connection”, NY_DS, nydb1
    ```
  - 若要验证到 `NY_DS.nydb1` 的连接的 **dsi_compile_enable** 是否设置为 **on**，请输入：

    ```
    admin config, “connection”, NY_DS, nydb1,dsi_compile_enable
    ```
  - 若要显示名称中包含“enable”的所有数据库级配置参数（如 **dsi_compile_enable**），请输入：

    ```
    admin config, “connection”, NY_DS, nydb1,"enable"
    ```

    注意： 必须用引号将“enable”引起来，因为它是 Replication Server 中的保留字。请参见《Replication Server 参考手册》的“主题”中的“保留字”。
- 表级：

  若要在使用 **dsi_command_convert** 在 NY_DS 数据服务器的 `nydb1` 数据库中的 `tb1` 表上设置 **d2none** 后显示所有配置参数，请输入：

  ```
  admin config, “table”, NY_DS, nydb1
  ```

请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**admin config**”。

显示表引用

可以使用在 Replication Server 系统数据库 (RSSD) 上执行的 **rs_helprep** 查看有关表引用的信息和 RTL 信息。

若要显示有关使用 **create replication definition** 创建的 **authors_repdef** 复制定义的信息，请输入：

```
rs_helprep authors_repdef
```

请参见《Replication Server 参考手册》的“RSSD 存储过程”中的“**rs_helprep**”。

Replication Server 中的系统表支持

Replication Server 使用 `rs_tbconfig` 表来存储支持表级配置参数，并使用 `rs_columns` 表中的 `ref_objowner` 和 `ref_objname` 列来支持参照约束。

有关完整表说明，请参见《Replication Server 参考手册》中的“Replication Server 系统表”。

混合版本支持和向后兼容性

仅当外发路由版本高于 15.5 时，HVAR 才能复制在复制定义中指定的参照约束。

如果出站路由版本早于 15.5，HVAR 可运行。不过，没有任何参照约束信息可供 Replication Server 15.5 版或更高版本使用。

连续复制是可供所有受支持版本的 Replication Server 使用的缺省复制模式。HVAR 仅可供 Replication Server 15.5 及更高版本使用。

提高的 DSI 效率

启用 **dsi_cmd_prefetch** 可缩短数据复制延迟，缩短数据复制延迟减少了 Replication Server 通过 **ct_results** 例程等待复制数据服务器的结果的时间长度，随后减少了数据服务器等待 Replication Server 的时间长度。

dsi_cmd_prefetch 的工作方式如下：

- 允许 Replication Server 在处理来自复制数据服务器的当前批处理的结果之前为复制数据服务器准备下一批处理命令。
- 提高了 DSI 执行程序 (DSI/E) 和 DSI 调度程序 (DSI/S) 线程之间的并发性。

使用 **alter connection** 或 **create connection** 将 **dsi_cmd_prefetch** 设置为 on。

例如，若要为到 SYDNEY_DS 数据服务器中的 `pubs2` 数据库的连接启用 **dsi_cmd_prefetch**，请输入：

```
alter connection to SYDNEY_DS.pubs2
set dsi_cmd_prefetch to 'on'
```

缺省值：off。

dsi_cmd_prefetch 是动态参数。在启用参数以使更改生效后，无需挂起并恢复数据库连接。

如果您还调优数据服务器以增强性能，在启用 **dsi_cmd_prefetch** 时，可能会进一步提高性能。

注意： 在将 **dsi_compile_enable** 设置为“on”时，Replication Server 会忽略您为 **dsi_cmd_prefetch** 设置的内容。

提高的 RepAgent 执行程序线程效率

在 RepAgent 执行程序线程进行分析时，可通过并行使用 NRM 线程规范化并打包日志传送语言 (LTL) 命令来提高 Replication Server 性能。

在指示 Replication Server 启用 NRM 线程时，将从 RepAgent 执行程序线程中拆分出一个线程以变为 NRM 线程。NRM 线程进行的并行处理减少了 RepAgent 执行程序线程的响应时间。

在启用 NRM 线程后，您可以指定可供消息队列中的 RepAgent 执行程序线程和 NRM 线程使用的内存。

启用 NRM 线程

可以使用 **configure replication server** 将 **nrm_thread** 设置为 on 以启用 NRM 线程。

输入：

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

缺省值：off

nrm_thread 是服务器级参数。在更改该参数值后，必须重新启动 Replication Server。

指定可供 RepAgent 执行程序使用的内存

在将 **nrm_thread** 设置为 on 后，请使用带 **exec_nrm_request_limit** 参数的 **configure replication server** 或 **alter connection** 指定可供消息队列中的 RepAgent 执行程序线程和 NRM 线程使用的总内存量。

如果消息队列上的命令使用的总内存量大于使用 **exec_nrm_request_limit** 指定的内存量，RepAgent 执行程序线程将休眠并等待内存变为可用。当 NRM 线程处理消息队列上的命令时，它将为 RepAgent 执行程序线程释放内存。

例如，要将到 SYDNEY_DS 数据服务器上的 `pubs2` 数据库的连接的 **exec_sqm_nrm_request_limt** 设置为 1GB，请输入：

```
alter connection to SYDNEY_DS.pubs2
set exec_nrm_request_limit to '1073741824'
```

exec_nrm_request_limit 值：

- 缺省值：
  - 32 位 – 1,048,576 字节 (1MB)
  - 64 位 – 8,388,608 字节 (8MB)
- 最大值 – 2,147,483,647 bytes (2GB)
- 最小值 – 16,384 字节 (16KB)

在更改 **exec_nrm_request_limit** 的配置后，挂起并恢复 Replication Agent。要挂起和恢复：

- RepAgent for Adaptive Server，请在 Replication Server 中执行 **sp_stop_rep_agent**，然后执行 **sp_start_rep_agent**。
- Replication Agent（针对支持的非 ASE 数据库），请在 Replication Agent 中执行 **suspend**，然后执行 **resume**。

提高的分配器线程读取效率

让分配器 (DIST) 线程能直接从稳定队列事务线程 (SQT) 高速缓存中读取 SQL 语句。这减少了 SQT 的负载以及二者之间的依赖性，并提高了 SQT 和 DIST 的效率。

可以使用带 **dist_direct_cache_read** 参数的 **configure replication server** 以使用此增强功能：

输入：

```
configure replication server
set dist_direct_cache_read to ‘on’
```

缺省情况下，**dist_direct_cache_read** 设置为“off”。如果禁用该参数，分配器线程将通过消息队列从 SQT 请求 SQL 语句。这会导致入站和出站队列争用。

dist_direct_cache_read 是服务器级参数。在启用或禁用该参数后，必须重新启动 Replication Server。

增强的内存分配

可以使用带 **mem_reduce_malloc** 参数的 **configure replication server** 在 Replication Server 中以更大的块形式分配内存。

这可降低所需的内存分配次数，从而提高 Replication Server 性能。

输入：

```
configure replication server
set mem_reduce_malloc to ‘on’
```

缺省情况下，**mem_reduce_malloc** 设置为“off”。

mem_reduce_malloc 是动态参数。在更改参数设置时，无需挂起或恢复数据库连接。

增加队列块大小

增加队列块大小以提高复制性能。

队列块大小是稳定队列结构使用的连续内存块中的字节数。设置较大的队列块大小可使 Replication Server 在单个块中处理更多的事务。您可以将队列块大小从缺省值 16KB 增加到 32KB、64KB、128KB 或 256KB。性能改进还依赖事务配置文件和环境。

注意： 您必须具有名为 REP_HVAR_ASE 的高级服务选件许可证才能使用增加队列块大小功能。

建议
Sybase 强烈建议您：

- 在增加队列块大小之前检验您是否具有足够的内存。
- 试用不同的队列块大小来确定适用于您的复制系统的最佳值。

限制

- 确保在进行队列块大小更改时，没有数据流入 Replication Server 中。
- 在正在实现预订时、正在取消实现时或正在创建或破坏路由时，您无法更改队列块大小。在 Replication Server 继续运行时，队列块大小更改将终止，并出现错误消息。
- 在您启动过程来更改队列块大小后，Replication Server 不会在第一个更改完成之前接受其它更改队列块大小的命令。
- 不要使用任何其它过程在 RSSD 中直接更改队列块大小，因为这些过程可能导致队列块大小配置不一致并引发 Replication Server 关闭。

注意： 在块大小更改后，清除所有队列。

更改队列块大小
修改队列块大小是对 Replication Server 配置的重大更改，将影响与 Replication Server 的所有连接。您必须挂起日志传送并停顿 Replication Server。

在队列块大小更改过程中，“上游”是指为您要更改其中的队列块大小的 Replication Server 馈送数据的所有复制系统组件，而“下游”是指从受影响的 Replication Server 接收数据的组件。

1. 若要保持数据完整性，请在更改队列块大小之前，阻止数据流入要配置的 Replication Server 中。
   a) 挂起从所有 Replication Agent 到您要配置的 Replication Server 的日志传送。
   b) 挂起来自 Replication Agent 的所有上游日志传送。
   c) 停顿所有上游 Replication Server。
   d) 挂起到您要配置的 Replication Server 的所有传入路由。
   e) 停顿您要配置的 Replication Server。
2. 将 **configure replication server** 与 **set block_size to** '*value*' 子句一起使用以设置要配置的 Replication Server 上的队列块大小。
   此命令将：
   - 检验是否没有正在进行预订实现。
   - 检验是否已挂起所有日志传送。
   - 检验是否已挂起所有传入路由。
   - 检验 Replication Server 是否已停顿。
   - 清除队列。

- 将 RSSD `rs_locater` 系统表中的值设置为零，以允许 Replication Agent 重新发送在您启动队列块大小更改过程时可能未应用于复制数据库的事务。
- 将队列块大小设置为您输入的值。
- （可选）如果包括 **with shutdown** 选项，则会关闭 Replication Server。在重新启动 Replication Server 时，队列块大小更改生效。关闭可确保 Replication Server 清除所有内存。

3. 在将队列块大小更改为较大的值后，删除或删除并重新创建您使用较小的块大小值创建的原始分区。

   只有在更改块大小后创建分区时，分区才会注册正确的段数。

4. 恢复数据流：
   a) 如果使用 **with shutdown** 选项，请重新启动 Replication Server。
   b) 从 Replication Agent 重新开始日志传送。
   c) 重新开始所有传入路由。
5. 检查所有下游 Replication Server RSSD 和数据服务器上的数据丢失。通常，您配置的 Replication Server 的 RSSD 中存在数据丢失。忽略从配置的 Replication Server 的 RSSD 接收数据的复制 RSSD 中的数据丢失。

   按照过程修复数据服务器中的数据丢失。如果 RSSD 中发生数据丢失，则会在受影响的 Replication Server 的日志中看到类似以下内容的消息：

   ```
   E. 2010/02/12 14:12:58. ERROR #6067 SQM(102:0 primaryDS.rssd) - /
   sqmoqid.c(1071)
   Loss detected for replicateDS.rssd from primaryDS.RSSD
   ```

   replicateDS 是复制数据服务器名称，*primaryDS* 是主数据服务器名称。

增加简单复制系统中的队列块大小

在简单复制系统中设置主 Replication Server 和复制 Replication Server 的队列块大小。

复制系统包括：

- 主数据库 – `pdb`
- 复制数据库 – `rdb`
- 主 Replication Server – PRS
- 主 Replication Server 的 RSSD – `pRSSD`
- 复制 Replication Server – RRS
- 复制 Replication Server 的 RSSD – `rRSSD`

```
          pRSSD       rRSSD

            |           |

pdb -----> PRS -----> RRS -----> rdb
```

在此示例中，RSSD 将 Adaptive Server 称为 Replication Server 系统数据库 (RSSD)，将 SQL Anywhere® 称为嵌入式 Replication Server 系统数据库 (ERSSD)。有关所有命令的完整语法、示例和用法信息，请参见《Replication Server 参考手册》。

1. 配置主 Replication Server：
   a) 挂起来自所有 Replication Agent 的日志传送。在主 Replication Server 上，执行：
   ```
   suspend log transfer from all
   ```
   b) 停顿主 Replication Server：
   ```
   admin quiesce_force_rsi
   ```
   c) 将主 Replication Server 上的队列块大小设置为 64KB：
   ```
   configure replication server
   set block_size to '64'
   ```
   （可选）使用 **with shutdown** 选项设置块大小并关闭主 Replication Server。例如：
   ```
   configure replication server
   set block_size to '64' with shutdown
   ```
   d) 查看事务日志以验证主 Replication Server 是否正在实现，日志传送和路由是否已挂起，以及主 Replication Server 是否已停顿。
   e) 重新启动主 Replication Server（如果您关闭了它）。请参见《Replication Server 管理指南第一卷》的“管理复制系统”中的“启动 Replication Server”。
   f) 查看主 Replication Server 事务日志以验证块大小是否已更改。
   g) 重新开始传送日志以允许 Replication Agent 连接到主 Replication Server。在主 Replication Server 上，执行：
   ```
   resume log transfer from all
   ```
   h) 检查复制 Replication Server 日志文件以获取有关数据丢失的信息。通过在复制 Replication Server 上执行 **ignore loss** 命令，忽略从主 Replication Server RSSD 传送到复制 Replication Server RSSD 时发生的数据丢失：
   ```
   ignore loss from PRS.pRSSD to RRS.rRSSD
   ```
2. 配置复制 Replication Server：
   a) 挂起来自所有 Replication Agent 的日志传送。在主 Replication Server 上和复制 Replication Server 上，执行：
   ```
   suspend log transfer from all
   ```
   b) 停顿主 Replication Server：
   ```
   admin quiesce_force_rsi
   ```
   c) 在发起到复制 Replication Server 的路由的所有 Replication Server 上，挂起路由：
   ```
   suspend route to RRS
   ```
   d) 停顿复制 Replication Server：
   ```
   admin quiesce_force_rsi
   ```
   e) 将复制 Replication Server 上的块大小设置为 64KB：
   ```
   configure replication server
   set block_size to '64'
   ```
   （可选）使用 **with shutdown** 选项关闭复制 Replication Server。例如：

```
configure replication server
set block_size to '64' with shutdown
```

f) 查看事务日志来验证复制 Replication Server 是否正在实现，日志传送和路由是否已挂起，以及复制 Replication Server 是否已停顿。
g) 重新启动复制 Replication Server（如果您关闭了它）。
h) 查看复制 Replication Server 事务日志以验证块大小是否已更改。
i) 重新开始传送日志以允许 Replication Agent 连接到复制 Replication Server。在复制 Replication Server 上，执行：

```
resume log transfer from all
```

j) 重新开始传送日志以允许 Replication Agent 连接到主 Replication Server。在主 Replication Server 上，执行：

```
resume log transfer from all
```

k) 重新开始挂起的路由：

```
resume route to RRS
```

l) 检查主和复制 Replication Server 日志文件以获取有关数据丢失的信息。如果将复制 RSSD 复制到主 RSSD，可通过在主 Replication Server 上执行 **ignore loss** 命令，忽略在主 RSSD 和复制 RSSD 之间发生的数据丢失。

```
ignore loss from RRS.rRSSD to PRS.pRSSD
```

另请参见

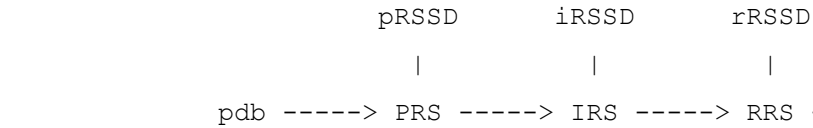
- 恢复丢失 （第 316 页）

增加具有中间路由的复制系统中的队列块大小

在具有中间路由的复制系统中设置主 Replication Server 的队列块大小。

复制系统包括：

- 主数据库 – pdb
- 复制数据库 – rdb
- 主 Replication Server – PRS
- 主 Replication Server 的 RSSD – pRSSD
- 复制 Replication Server – RRS
- 复制 Replication Server 的 RSSD – rRSSD
- 中间 Replication Server – IRS
- 中间 Replication Server 的 RSSD – iRSSD



在此示例中，RSSD 将 Adaptive Server 称为 Replication Server 系统数据库 (RSSD)，将 SQL Anywhere 称为嵌入式 Replication Server 系统数据库 (ERSSD)。有关所有命令的完整语法、示例和用法信息，请参见《Replication Server 参考手册》。

1. 挂起来自所有 Replication Agent 的日志传送。在主 Replication Server 上，执行：
   ```
   suspend log transfer from all
   ```
2. 停顿 PRS：
   ```
   admin quiesce_force_rsi
   ```
3. 将主 Replication Server 上的块大小设置为 64KB：
   ```
   configure replication server
   set block_size to '64'
   ```
   （可选）使用 **with shutdown** 选项设置块大小并关闭主 Replication Server。例如：
   ```
   configure replication server
   set block_size to '64' with shutdown
   ```
4. 查看事务日志以验证主 Replication Server 是否正在实现，日志传送和路由是否已挂起，以及主 Replication Server 是否已停顿。
5. 重新启动主 Replication Server（如果您关闭了它）。请参见《Replication Server 管理指南第一卷》的“管理复制系统”中的“启动 Replication Server”。
6. 查看主 Replication Server 事务日志以验证块大小是否已更改。
7. 重新开始传送日志以允许 Replication Agent 连接到主 Replication Server。在主 Replication Server 上，执行：
   ```
   resume log transfer from all
   ```
8. 检查中间和复制 Replication Server 日志文件以获取有关数据丢失的信息。通过在中间 Replication Server 上执行两次 **ignore loss** 命令，忽略从主 Replication Server RSSD 传送到复制 Replication Server 以及从主 RSSD 传送到中间 RSSD 时发生的数据丢失：
   ```
   ignore loss from PRS.pRSSD to RRS
   go
   ignore loss from PRS.pRSSD to IRS.iRSSD
   ```

另请参见

- 恢复丢失 （第 316 页）

Multi-Path Replication

可以使用多个复制路径来提高复制吞吐量和性能并减少争用。

在单路径复制环境中，以串行方式将事务从主数据库复制到复制数据库以确保遵循主数据库事务提交顺序，从而确保复制数据库与主数据库保持一致。即使多个应用程序通常以并行方式在主数据库中执行各自的事务，或者事务来自多个主数据库，将事务应用于复制数据库的串行模式也会保持不变。

某些复制环境可以在一个表子集中保持数据一致性，而无需将来自同一主数据库的所有事务都进行串行化。这种环境的一个典型示例是，当访问不同数据集的不同应用程序修改单个主数据库时。包含特定应用程序修改的表子集的不同数据集继续以串行方式进行复制。不同表子集中的数据可以按并行方式进行复制。

Multi-Path Replication™ 支持通过不同的流复制数据，同时仍在路径中保持数据一致性，但不会在不同的路径中遵循提交顺序。

复制路径包含 Replication Server 与主数据库或复制数据库之间的所有组件和模块。在多路径复制中，可以为多个 Replication Agent 连接创建从主数据库到一个或多个 Replication Server 的多个主复制路径，以及创建从一个或多个 Replication Server 到复制数据库的多个复制复制路径。可以在热备份和多节点可用性 (MSA) 环境中配置多路径复制。您可以在 Replication Server 之间的专用路由上传送事务以避免共享路由上发生拥塞，并且可以将通过 Replication Server 从主数据库到复制数据库的端对端复制路径专用于表和存储过程等对象。

许可证

Multi-Path Replication 是作为高级服务选件的一部分许可的。请参见《Replication Server 安装指南》的“规划安装”中的“获取许可证”。

系统要求

Replication Server 支持 Adaptive Server 数据库之间的多路径复制，其中主数据服务器是 Adaptive Server 15.7 和更高版本。对于多路径复制系统中的非 Adaptive Server 数据库，请参见《Replication Server 异构复制指南》中的：

- “Sybase IQ 作为复制数据服务器”中的“到 Sybase IQ 的多路径复制”。
- “异构多路径复制”。

Multi-Path Replication 快速入门

为端对端复制设置一个由两个主和复制路径组成的多路径复制复制系统。

1. 选择或创建两组您要通过两个复制路径复制的表或存储过程。
2. 使用 **rs_init** 向复制系统中添加主和复制 Adaptive Server 数据库。
3. 启用多线程 RepAgent。
   在主 Adaptive Server 上，输入：

   ```
   sp_config_rep_agent primary_database_name, 'multithread rep
   agent', 'true'
   ```

4. 为 RepAgent 设置复制路径数。
   例如，若要启用两个路径，请输入：

   ```
   sp_config_rep_agent primary_database_name, 'max number of
   replication paths', '2'
   ```

5. 创建从主数据库到 Replication Server 的替代复制路径。
   a) 创建名为 *alternate_path_name* 的替代物理 RepAgent 复制路径。
   在主 Adaptive Server 上，输入：

   ```
   sp_replication_path "primary_database_name", 'add',
   "alternate_path_name", "repserver_name",
   "repserver_user", "repserver_password"
   ```

b) 创建从 Replication Server 到主数据库的对应替代主连接并通过使用同一 RepAgent 复制路径名 *alternate_path_name* 将其绑定到替代物理 RepAgent 复制路径。
在 Replication Server 上输入：

```
create alternate connection to
primary_dataserver.primary_database
named primary_dataserver.alternate_path_name
set error class to rs_sqlserver_error_class
set function string class to rs_sqlserver_function_class
set username to primary_db_maintenance_user
set password to primary_db_maintenance_password
with primary only
```

复制系统包含两个主复制路径：缺省值和 *alternate_path_name*

6. 使用同一替代复制路径名 *alternate_path_name* 创建从 Replication Server 到复制数据库的替代复制连接。

```
create alternate connection to
replicate_dataserver.replicate_database
named replicate_dataserver.alternate_path_name
```

复制系统包含两个复制复制路径：缺省值和 *alternate_path_name*

7. 将一组对象（如表或存储过程）绑定到替代复制路径。

```
sp_replication_path pdb, 'bind', "table",
"[table_owner].table_name", "alternate_path_name"
```

另一组对象使用缺省复制路径。只能将对象绑定到替代复制路径。所有您不想绑定到替代复制路径的对象都使用缺省路径。

8. 针对主数据库创建复制定义。
例如，为 `authors` 表创建 **authors_rep** 复制定义：

```
create replication definition authors_rep
with primary at primary_dataserver.primary_database
with all tables named 'authors'
...
go
```

如果缺省主连接和替代主连接位于不同 Replication Server 上，请在每个 Replication Server 上都创建复制定义。

9. 针对缺省主连接和缺省复制连接创建预订。

```
create subscription subscription_default_path for
replication_definition
with primary at primary_dataserver.primary_database
with replicate at replicate_dataserver.replicate_database
```

10. 针对替代主连接和替代复制连接创建预订。

```
create subscription subscription_alternate_path for
replication_definition
with primary at primary_dataserver.alternate_path_name
with replicate at replicate_dataserver.alternate_path_name
```

并行事务流

只要事务可以分成并行流，而且不必跨不同流串行提交，多路径复制就可改进复制性能。

可以通过将事务分成并行复制路径以减少拥挤来改进性能。可以根据并行化规则（如事务属性或派生数据值）来划分事务。例如，您可以执行以下操作：

- 指定对象（如表或存储过程）的专用路径。当您将对象绑定到路径时，Replication Agent 会通过该路径将您对该对象执行的所有复制操作发送到您在多个复制路径配置中定义的 Replication Server。RepAgent for Adaptive Server 和 Replication Agent for Oracle 支持此复制分布模式。
- 按主数据库上的客户端连接的会话 ID 划分事务。RepAgent for Adaptive Server 按支持客户端连接分布事务。
- 每个路径专用一个 Replication Server。
- 通过将对象绑定到专用路径或较不拥挤的路径或通过在 Replication Server 之间创建专用路由来分配这些路径。

多路径复制的分配模式

在 Multi-Path Replication™ 环境中，可通过来自主数据库的可用主复制路径分配来自该数据库的复制负载，使用不同的分配模式实现并行复制并提高复制性能。

通过使用多个复制路径，您可以选择下列分配模式之一：

- **按对象绑定分配** – 将对象（如表和存储过程）绑定到特定复制路径以并行启用这些对象的复制。
- **按连接分配** – Adaptive Server RepAgent 将来自不同客户端进程的事务分配给可用的复制路径。

缺省模式是按对象绑定分配。RepAgent 不支持一次使用多种分配模式。

另请参见

- 设置分布模式 （第 235 页）

按对象绑定分配

在 Multi-Path Replication 环境中，您可以将对象（如表或存储过程）绑定到一个或多个主复制路径。

在将对象绑定到路径时，RepAgent 通过此路径将对该对象执行的任何可复制的操作发送到多复制路径配置中定义的 Replication Server。如果未将对象绑定到路径，则 RepAgent 使用缺省路径将该对象发送到缺省路径中定义的 Replication Server。无法将对象绑定到缺省路径，并且无需执行任何操作即可通过缺省路径发送对象。在复制过程中，绑定对象始终遵循相同的路径。

数据库支持
Replication Server 支持来自 Adaptive Server 15.7 和更高版本的主数据库的多路径复制使用按对象绑定分配。

按连接分配
在 Multi-Path Replication 环境中，您可以通过来自主数据库的可用主复制路径分配来自该数据库的多个客户端连接的复制负载。

到主数据库的每个客户端连接和每个服务器进程具有唯一的系统进程 ID (spid)。Adaptive Server RepAgent 使用数据库事务日志中存储的 spid 值来识别特定客户端连接或服务器进程完成的事务。如果客户端断开连接并重新连接，客户端的 spid 将会发生变化。

对于按连接分配，Adaptive Server RepAgent 将来自不同客户端进程的事务分配给可用的复制路径。随着时间的推移，数据分配趋于在所有可用的路径之间达到平衡。如果有更多的可用 RepAgent 路径，而且客户端进程数很大，则会提高复制性能并且复制负载分配更加平均。

如果配置了按连接分配，则 RepAgent 根据 spid 和可用复制路径数通过复制路径分配事务。特定 spid 完成的所有事务始终使用相同的复制路径。随着时间的推移，分配给每个路径的 spid 数将变得大致相同，因为 spid 分配趋于平均。假设有三个用户，用户 1 的 spid 为 0，用户 2 为 1，用户 3 为 5，并且有三个复制路径：缺省路径为 `PDS.pdb1_conn1` 和 `PDS.pdb1_conn2`。在此示例中：

- 用户 1 spid 0 完成的所有事务是通过缺省连接复制的。
- 用户 2 spid 1 完成的所有事务是通过 `PDS.pdb1_conn1` 连接复制的。
- 用户 3 spid 5 完成的所有事务是通过 `PDS.pdb1_conn2` 连接复制的。

用户 1 生成的事务无法通过 `PDS.pdb1_conn1` 或 `PDS.pdb1_conn2` 进行复制，用户 2 生成的事务无法通过缺省连接或 `PDS.pdb1_conn1` 连接进行复制。

如果具有新的可用路径，并且未重新启动 RepAgent 以识别新路径，则 RepAgent 继续将事务分配给现有的路径。在重新启动 RepAgent 时，RepAgent 重新装载所有路径的配置，并且路径数量和目标更改将会生效。可以通过不同的路径发送事务，或将其传送到不同的目标 Replication Server。不过，如果事务使用不同的路径，则可能会出现重复的事务。如果 RepAgent 使用按连接分配，Sybase 建议您不要更改来自主数据服务器的复制路径的目标 Replication Server。

在启用按连接分配后，RepAgent 不使用绑定，并且绑定没有任何作用。在使用 **sp_replication_path** 列出定义的任何对象绑定时，您仍然可以看到这些绑定。在更改以下内容时，必须重新启动 RepAgent 并停顿 Replication Server：

- 分配模式 - 如果当前分配模式为连接，当 RepAgent 在启动期间装载复制路径时，RepAgent 不会处理任何对象绑定。

- 复制拓扑 – 如果未重新启动 RepAgent 并停顿 Replication Server，可能会导致数据丢失或数据重复。

数据库支持
Replication Server 支持来自 Adaptive Server 15.7 ESD #1 和更高版本的主数据库的多路径复制使用按连接分配。

按连接分配的限制
按连接分配模式存在一些限制。

- 按连接分配不会以串行方式在连接之间发送事务。与主数据库日志中的事务顺序相比，一个连接上的事务到达复制数据库时相对于另一个连接上的相邻事务的顺序可能会有所不同。
- 按连接分配不保证事务分配保持平均和负载平衡：
  - 如果某些 RepAgent 发送器上的大事务不成比例地下降，则其它发送器线程可能会填满。
  - 某个 RepAgent 发送器可能很忙，并且仍有事务进入该发送器队列。
- 按连接分配模式使用与用户或应用程序会话关联的 SPID 在所有路径之间分配复制的数据。只要与用户会话关联的 SPID 保持不变，该用户或应用程序的所有事务就会使用相同的路径。如果用户或应用程序直接连接到与 RepAgent 相同的数据服务器上，则 SPID 不会改变，并且不会出现事务序列化问题。

  不过，如果用户或应用程序使用具有连接池的中间层应用程序服务器在 RepAgent 连接到的数据服务器上执行事务，则用于执行事务的 SPID 可能会发生变化。例如，如果来自用户的 **insert** 事务后跟相同数据的 **update** 事务，则 **insert** 事务可能具有与 **update** 事务不同的 SPID。发生这种情况可能是因为，中间层应用程序服务器可能使用不同的连接来执行每个事务。在这种情况下，RepAgent 在不同的路径上发送 **update** 和 **insert**；如果 **update** 在 **insert** 之前到达复制数据库，则可能会出现事务序列化问题。

缺省和替代连接

在多路径复制中，连接包括缺省连接以及一个或多个替代连接。

从 Replication Agent 接受数据的连接是主连接，将数据应用到数据库的连接是复制连接。缺省或替代连接可以是主连接或复制连接。

缺省连接是当您向复制域中添加数据库时创建的从 Replication Server 到特定主数据库或复制数据库的连接。可以根据数据服务器是 Adaptive Server 还是支持的非 ASE 数据服务器，使用 **rs_init**、Replication Manager Sybase Central 插件、**create connection** 或 **create connection ... using profile** 创建缺省连接。

缺省连接以 *dataserver.database* 形式的连接名称使用数据服务器和数据库名称，其中 *dataserver* 和 *database* 分别是数据服务器和数据库名称。

可以在创建缺省连接（必需）后创建多个替代连接。每个替代连接都必须具有唯一的名称。

创建替代连接后，可以更改连接属性或删除连接。还可以显示所有连接的状态以及为连接创建预订。

当您创建替代连接时，用户 ID 必须是有效的用户。当您创建到 Sybase IQ 复制数据库的连接时，必须为缺省连接和每个替代连接指定连接配置文件与连接配置文件版本以及唯一的维护用户名。

到复制数据库的多个连接

创建从 Replication Server 到复制数据库的多个连接。

在创建到复制数据库的多个连接时，每个复制连接将预订一个事务流。同一个流中的事务遵循主提交顺序。不同流中的事务是并行应用的，可能不遵循主提交顺序。

缺省和替代复制连接

还可以创建从多个 Replication Server 到相同复制域中的相同复制数据库的复制连接。复制域中只有一个 Replication Server 可以拥有和控制缺省复制连接。无法创建来自域中的其它 Replication Server 的多个缺省复制连接。其它 Replication Server 只能具有替代复制连接。

在创建替代复制连接后，您可以修改连接属性或删除连接。还可以显示所有连接的状态以及为连接创建预订。

创建替代复制连接

可以使用 **create alternate connection** 创建从 Replication Server 到复制数据库的替代连接。

输入：

```
create alternate connection to dataserver.database
named conn_server.conn_db
[set error class [to] error_class
set function string class [to] function_class
set username [to] user
set password [to] pwd]
[set database_param [to]  'value' ]
```

其中：

- *dataserver* 和 *database* – 是复制数据服务器和数据库。
- *conn_server.conn_db* – 替代复制连接，由数据服务器名称和连接名称组成。
  - 如果 *conn_server* 不同于 *dataserver*，则 interface 文件中必须包含 *conn_server* 的条目。
  - 如果 *conn_server* 与 *dataserver* 相同，则 *conn_db* 必须不同于 *database*。
  - 每个复制连接名称在复制系统中必须是唯一的。
- **set function string class [to]** *function_class*、**set username [to]** *user* 和 **set password [to]** *pwd* – 可以在创建替代连接时使用的 **alter connection** 和 **create connection** 的现有子句。
  - 如果忽略这些子句，替代复制连接将继承您对缺省复制连接设置的值。

  - 如果在与控制缺省连接的（控制器）Replication Server 不同的（当前）Replication Server 上创建替代连接时忽略这些子句，当前 Replication Server 会返回错误。
  - 仅当同一个 Replication Server 同时控制替代连接和缺省连接时，替代连接才可以继承缺省连接中的值。
  - 如果不为替代连接设置维护用户，该连接将继承缺省连接的维护用户。替代连接将使用您为替代连接指定的任何新维护用户。
- *set param* – **alter connection** 和 **create connection** 的现有可选连接参数的子句。
  - 为替代复制连接设置的任何值都将覆盖从缺省连接继承的值或缺省值。
  - 仅当同一个 Replication Server 同时控制替代连接和缺省连接时，替代连接才可以继承缺省连接中的值。

例如，要创建到 FINANCE_DS 数据服务器上的 rdb 复制数据库的名为 FINANCE_DS2.rdb_conn2 的替代复制连接，请输入：

```
create alternate connection to FINANCE_DS.rdb
named FINANCE_DS2.rdb_conn2
go
```

注意： 您必须在 interfaces 或 sql.ini 文件中定义 FINANCE_DS 和 FINANCE_DS2。

更改或删除替代复制连接

可以使用 **alter connection** 和 **drop connection** 命令更改或删除缺省或替代复制连接。

在命令中指定的数据服务器和数据库名称可以是缺省或替代复制连接名称。

在配置替代或缺省复制连接时，可以使用适用于 **alter connection** 的配置参数。

例如，要将 TOKYO_DS.rdb_conn2 替代复制连接的 **dsi_max_xacts_in_group** 设置为 40，请输入：

```
alter connection to TOKYO_DS.rdb_conn2
set dsi_max_xacts_in_group to '40'
go
```

显示复制连接信息

可以将 **replicate** 参数与 **admin show_connections** 一起使用以显示有关所有复制连接的信息。

例如，在控制 FINANCE_DS 和 NY_DS 数据服务器中的复制数据库的 Replication Server 中，输入：

```
admin show_connections, 'replicate'
```

您会看到：

```
Connection Name         Server       Database   User
-------------------     ----------   --------   ----------
FINANCE_DS.fin_rdb      FINANCE_DS   fin_rdb    rdb_maint
NY_DS.ny_rdb_conn2      NY_DS        ny_rdb     rdb_maint
```

FINANCE_DS.fin_rdb 是 Replication Server 和 FINANCE_DS 数据服务器的 fin_rdb 数据库之间的缺省连接，因为连接名与数据服务器和数据库名的组合相匹配。

NY_DS.ny_db_conn2 是 Replication Server 和 NY_DS 数据服务器的 ny_rdb 数据库之间的替代连接，因为连接名与数据服务器和数据库名的组合不匹配。

(可选）使用 rs_databases 系统表列出到 Replication Server 的缺省连接和替代连接。

创建具有多个复制连接的复制系统

创建缺省复制连接和替代复制连接，然后创建相应的预订以构建多复制连接复制系统。

前提条件

确保可以并行运行事务，然后将复制事务分成两个组。

过程

可以将此示例方案作为创建具有多个复制连接的复制系统的范例，此方案包含 PDS 主数据服务器上的 pdb 主数据库，其中的 T1 和 T2 表具有相应的 **repdef1** 和 **repdef2** 复制定义。每个表具有一个影响它的事务集。相应的预订是 **sub1** 和 **sub2**。rdb 复制数据库位于 RDS 复制数据服务器上，主 Replication Server 和复制 Replication Server 分别是 RS1 和 RS2。

1. 在 RS1 中，使用 **rs_init** 或 **create connection** 创建到复制数据库的缺省复制连接。

   ```
   create connection to RDS.rdb
   using profile ase_to_ase;standard
   set username to rdb_maint
   set password to rdb_maint_ps
   go
   ```

2. 在 RS1 中，创建到 rdb 复制数据库的名为 RDS.rdb1 的替代复制连接。

   ```
   create alternate connection to RDS.rdb
   named RDS.rdb1
   go
   ```

   (可选）从 RS2 中创建到复制数据库的另一个替代复制连接。在 RS2 中，输入：

   ```
   create connection to RDS.rdb
   named RDS.rdb2
   set error class to rs_sqlserver_error_class
   set function string class to rs_sqlserver_function_class
   set username to rdb_maint
   set password to rdb_maint_ps
   go
   ```

3. 创建 **sub1** 预订，并指定缺省复制连接以复制第一个事务集中的事务。

   ```
   create subscription sub1 for repdef1
   with replicate at RDS.rdb
   go
   ```

4. 创建 **sub2** 预订，并指定替代复制连接以复制第二个事务集中的事务。

```
create subscription sub2 for repdef2
with replicate at RDS.rdb2
go
```

来自主数据库的多个连接

创建和管理从 Replication Server 到主数据库的多个连接，可以将这些连接与从主数据库到 Replication Server 的 RepAgent 路径相关联。

创建替代主连接

可以使用 **create alternate connection** 创建从 Replication Server 到主数据库的替代连接。

输入：

```
create alternate connection to dataserver.database
named conn_server.conn_db
[with {log transfer on | primary only}]
```

其中：

- *dataserver* 和 *database* – 是主数据服务器和数据库。
- *conn_server.conn_db* – 是替代主连接名称，由数据服务器名称和连接名称组成。
  - 如果 *conn_server* 与 *dataserver* 相同，则 *conn_db* 必须不同于 *database*。
  - *conn_server.conn_db* 必须与 Replication Agent 和 Replication Server 之间的连接名称相匹配。
  - 每个主连接名称在复制系统中必须是唯一的。
- **with log transfer on** – 指示 Replication Server 创建到 *dataserver.database* 中指定的数据库的替代主连接和替代复制连接，两个连接具有 *conn_server.conn_db* 中指定的名称。
- **primary only** – 指示 Replication Server 仅创建到主数据库的替代主连接，该连接具有 *conn_server.conn_db* 中指定的名称。

例如，要创建到 `SALES_DS` 数据服务器上的 `pdb` 数据库的名为 `SALES_DS.pdb_conn2` 的替代主连接，请输入：

```
create alternate connection to SALES_DS.pdb
named SALES_DS.pdb_conn2
with primary only
go
```

更改或删除替代主连接

可以分别使用现有的 **alter connection** 和 **drop connection** 命令更改或删除缺省主连接或替代主连接。

例如，可以使用 **alter connection** 启用或禁用到 *dataserver.database* 中指定的主数据库的缺省主连接：

```
alter connection to dataserver.database
set primary only [on|off]
```

设置为 off 可启用复制连接。

显示主连接信息

可以将 **primary** 参数与 **admin show_connections** 一起使用以显示有关所有主连接的信息。

例如，在控制 SALES_DS 数据服务器中的主数据库的 Replication Server 中，输入：

```
admin show_connections, 'primary'
```

您会看到：

```
Connection Name     Server         Database     User
----------------    ----------     ---------    ----------
SALES_DS.pdb        SALES_DS       pdb          pdb_maint
SALES_DS.pdb_conn2  SALES_DS       pdb          pdb_maint
```

SALES_DS.pdb 是 Replication Server 和 SALES_DS 数据服务器的 pdb 数据库之间的缺省连接，因为连接名与数据服务器和数据库名的组合相匹配。

SALES_DS.pdb_conn2 是 Replication Server 和 SALES_DS 数据服务器的 pdb 数据库之间的替代连接，因为连接名与数据服务器和数据库名的组合不匹配。

(可选）使用 rs_databases 系统表列出到 Replication Server 的缺省连接和替代连接。

复制定义和预订

可以使用复制定义和预订在多个替代连接中定义复制。

替代连接的复制定义和预订

为主数据库创建的复制定义应用于控制复制定义的 Replication Server 和主数据库之间的所有主连接（缺省连接和替代连接）。因此，必须先删除主数据库的所有复制定义才能删除到主数据库的最后一个主连接逻辑连接。

使用 1570 版本的系统时，您只能对数据库创建复制定义和发布。为 **create replication definition** 命令的 **with primary at** 子句指定的名称必须是主数据库名称。

由于主数据库和 Replication Server 之间的所有主连接共享所有复制定义，因此必须在预订中指定哪个主连接是数据源，哪个复制连接是复制目标。请在 **create subscription** 的 **with primary** 和 **with replicate** 子句中指定相应的缺省连接或替代连接名称。如果未在 **with primary** 子句中指定连接名称，则 Replication Server 针对到主数据库的缺省主连接创建预订。

```
create subscription sub_name
for {table_repdef | func_repdef | publication pub |
database replication definition db_repdef}
with primary at data_server.database
with replicate at data_server.database
[where {column_name | @param_name}
       {< | > | >= | <= | = | &} value
[and {column_name | @param_name}
```

```
     {< | > | >= | <= | = | &} value]...]
[without holdlock | incrementally | without materialization]
[subscribe to truncate table]
[for new articles]
```

从不支持替代连接的 Replication Server 版本中升级时，将在升级的 Replication Server 中保留针对缺省主连接和缺省复制连接定义的所有预订。

示例 1 – 预订替代主连接

要针对到 LON_DS 主数据服务器的 LON_DS.pdb_conn2 替代主连接上的 **repdef_conn2** 复制定义创建 **sub_conn2** 预订（缺省复制连接为 NY_DS.rdb），请输入：

```
create subscription sub_conn2 for repdef_conn2
with primary at LON_DS.pdb_conn2
with replicate at NY_DS.rdb
without materialization
go
```

示例 2 – 预订替代复制连接

要为 NY_DS.rdb_conn2 替代复制连接上的 **repdef_conn2** 复制定义创建 **sub_conn2** 预订，请输入：

```
create subscription sub_conn2 for repdef_conn2
with replicate at NY_DS.rdb_conn2
without materialization
go
```

在连接之间移动预订

可以使用 **alter subscription** 在使用同一 Replication Server 的同一复制数据库的复制连接之间移动预订而无需重新实现。

在复制 Replication Server 上执行 **alter subscription**：

```
alter subscription sub_name
for {table_repdef|func_repdef|{{publication pub|
database replication definition db_repdef}
with primary at primary_dataserver_name.primary_database_name}}
move replicate from ds_name.db_name
to ds_name1.db_name1
```

其中，您可以将预订从 *ds_name.db_name* 复制连接移动到 *ds_name1.db_name1* 复制连接。

例如，要将 **rep1** 复制定义的 **sub1** 预订从 RDS.rdb1 连接移动到 RDS.rdb2 连接，请输入：

```
alter subscription sub1 for rep1
move replicate from RDS.rdb1
to RDS.rdb2
```

如果主 Replication Server 版本早于 1570，则无法使用 **alter subscription**，而必须删除并在所需的连接中重新创建预订。

要删除必须通过同一路径进行复制的多个预订，请挂起主连接的日志传送，并在移动所有预订后重新开始日志传送。

多个主复制路径

创建多个从主数据库到一个或多个 Replication Server 的主复制路径以提高复制吞吐量并避免争用，或者将数据路由到不同的 Replication Server。

每个主复制路径包含从主数据库到 Replication Server 的 RepAgent 路径以及从 Replication Server 到主数据库的关联主连接。您可以将对象（如表或存储过程）绑定到其中的一个或多个路径。

物理路径定义了将接收绑定到路径的数据的 Replication Server 以及连接到相同 Replication Server 的 RepAgent 发送器线程。可以使用相同的连接名称将从主数据库到 Replication Server 的 RepAgent 物理路径与从 Replication Server 到主数据库的相应连接相关联。

逻辑路径使用一个名称将一个或多个物理路径划分在一起，以便将数据分配给多个 Replication Server。如果需要将表复制到多个目标，则可以将表绑定到划分相关物理路径的逻辑路径，而不是将表绑定到每个目标的物理路径。

创建多个主复制路径

您可以创建多个从主数据库到一个或多个 Replication Server 的主复制路径。每个主路径包含 RepAgent 路径和关联的主连接。

1. 启用多线程 RepAgent 并为多个复制路径启用 RepAgent。
2. 使用 **rs_init** 创建缺省主连接和 RepAgent 复制路径。
3. 创建替代主复制路径，每个路径包含链接到替代 RepAgent 复制路径的替代主连接。
4. 为通过复制路径的复制设置分配模式。
   - 按连接分配。
   - 按对象绑定分配。

启用多线程 RepAgent 并为 RepAgent 启用多个路径

启用多线程 RepAgent，并将其配置为使用来自主数据库的其它路径。

缺省情况下，Adaptive Server RepAgent 包含单个线程，用于扫描主数据库日志、生成 LTL 并将 LTL 发送到 Replication Server。通过多线程 RepAgent，扫描和发送活动由单独的线程来执行。

如果使用多线程 RepAgent，则始终包含一个缺省路径以将数据发送到 Replication Server。在数据库上最初启用 RepAgent 时，RepAgent 将创建缺省路径。

多个复制路径只能处理 Adaptive Server 15.7 和更高版本生成的 SQL 语句复制事务。

1. 设置 RepAgent 的可用内存

在启用和配置多个 RepAgent 发送器线程之前，必须为 Adaptive Server 中的 RepAgent 线程提供足够大的内存。

2. 启用多线程 RepAgent

   启用或禁用多线程 RepAgent，从而针对 RepAgent 扫描程序和发送器活动使用单独的线程。

3. 设置发送缓冲区数

   设置多线程 RepAgent 的扫描程序和发送器任务可以使用的最大发送缓冲区数量。

4. 为 RepAgent 设置最大复制路径数

   设置允许 RepAgent 从主数据库中复制数据时使用的最大路径数。RepAgent 会为每个 RepAgent 路径生成一个 RepAgent 发送器线程。

5. 显示配置参数设置

   可以使用 Adaptive Server 存储过程显示 RepAgent 配置参数设置，以及有关 RepAgent 多线程和多路径状态的其它信息。

设置 RepAgent 的可用内存

在启用和配置多个 RepAgent 发送器线程之前，必须为 Adaptive Server 中的 RepAgent 线程提供足够大的内存。

专用于 Adaptive Server 中的 RepAgent 线程的内存池的缺省大小为 4096 页。

1. 显示当前 RepAgent 线程池大小和其它 RepAgent 线程参数的设置。在主 Adaptive Server 上，输入：

   ```
   sp_configure 'Rep Agent Thread administration'
   go
   ```

   您会看到：

   ```
   Group: Rep Agent Thread Administration

   Parameter       Default  Memory  Config  Run    Unit      Type
   Name                     Used    Value   Value

   enable rep      0        0       1       1      switch    dynamic
   agent threads

   replication     4096     8194    4096    4096   memory    dynamic
   agent memory                                    pages(2k)
   size
   ```

   此示例显示，**enable rep agent threads** 是可以打开或关闭的动态参数。对动态参数进行的更改不需要重新启动 RepAgent。

2. 更改 Adaptive Server 分配给 RepAgent 线程池的内存。
   例如，要将池大小设置为 8194 页，请在主 Adaptive Server 上输入：

   ```
   sp_configure 'replication agent memory size', 8194
   go
   ```

   您会看到：

```
Group: Rep Agent Thread Administration

Parameter       Default  Memory  Config  Run    Unit      Type
Name                     Used    Value   Value

replication     4096     16430   8194    8194   memory    dynamic
agent memory                                    pages(2k)
size
(1 row affected)
Configuration option changed. ASE need not be rebooted since the
option is dynamic.
Changing the value of 'replication agent memory size' to '8194'
increases the amount of memory ASE uses by 8236 K.
```

启用多线程 RepAgent

启用或禁用多线程 RepAgent，从而针对 RepAgent 扫描程序和发送器活动使用单独的线程。

登录到主 Adaptive Server 并输入：

```
sp_config_rep_agent dbname, 'multithread rep agent', {'true' |
'false'}
```

其中 *dbname* 是 Adaptive Server 主数据库。

设置为 true 以启用多线程 RepAgent。缺省值为 false。必须重新启动 RepAgent 以使更改生效。

设置发送缓冲区数

设置多线程 RepAgent 的扫描程序和发送器任务可以使用的最大发送缓冲区数量。

可以在启用多线程 RepAgent 时设置发送缓冲区数，甚至在完成为多路径复制启用和配置 RepAgent 的过程后进行设置。

在主 Adaptive Server 上，输入：

```
sp_config_rep_agent dbname, 'number of send buffers',
'num_of_send_buffers'
```

其中 *dbname* 是 Adaptive Server 主数据库。

例如，要将 pdb1 数据库的发送缓冲区数设置为 40，请输入：

```
sp_config_rep_agent pdb1, 'number of send buffers', '40'
```

number of send buffers 缺省值为 50 个缓冲区。您可以设置 1 到 MAXINT 值 (2,147,483,647) 之间的值。该参数是动态的；您不需要重新启动 RepAgent。

每个发送缓冲区具有相同的大小，您可以使用 **send buffer size** RepAgent 参数设置该大小。请参见《Replication Server 参考手册》的“Adaptive Server 命令和系统过程”中的“**sp_config_rep_agent**”。

为 RepAgent 设置最大复制路径数

设置允许 RepAgent 从主数据库中复制数据时使用的最大路径数。RepAgent 会为每个 RepAgent 路径生成一个 RepAgent 发送器线程。

在主 Adaptive Server 上，输入：

```
sp_config_rep_agent dbname, 'max number replication paths', 'max
number replication paths value'
```

其中 *dbname* 是 Adaptive Server 主数据库。

例如，要在 pdb1 数据库上将 *max number replication paths* 设置为 3，请输入：

```
sp_config_rep_agent pdb1, 'max number replication paths', '3'
```

对于未明确绑定到路径的所有复制对象，如果 **max number replication paths** 大于 1，则 RepAgent 继续使用缺省路径。

对于绑定到路径的复制对象，如果 **max number replication paths** 小于路径数，RepAgent 会报告错误并终止。

显示配置参数设置

可以使用 Adaptive Server 存储过程显示 RepAgent 配置参数设置，以及有关 RepAgent 多线程和多路径状态的其它信息。

- **sp_config_rep_agent** – 仅指定数据库名称以显示使用 **sp_config_rep_agent** 设置的参数设置
- **sp_help_rep_agent** – 要显示有关 RepAgent 状态的其它信息，请指定：
  - **send** – 显示已分配给 RepAgent 的发送缓冲区数。
  - **config** – 显示有关 RepAgent 多路径配置参数的信息。
  - **process** – 在启用 **multithread rep agent** 时，显示有关多个 RepAgent 进程的信息。
- **sp_who** – 显示有关在 Replication Server 中运行的 RepAgent 进程和线程的信息

有关 **sp_config_rep_agent** 和 **sp_help_rep_agent** 的信息，请参见《Replication Server 管理指南参考手册》中的“Adaptive Server 命令和系统过程”。

请参见《Adaptive Server 参考手册：过程》的“系统过程”中的“**sp_who**”。

为主数据库创建替代复制路径

可以将 **add** 参数与 **sp_replication_path** 一起使用，通过将 RepAgent 复制路径与来自 Replication Server 的主连接相关联，在主数据库和 Replication Server 之间创建替代物理路径。

前提条件

使用 **rs_init** 在主数据库和 Replication Server 之间创建缺省复制路径。

过程

通过使用包含 PDS 主数据服务器、pdb 数据库以及 RS1 和 RS2 Replication Server 的示例复制系统，在主数据库上创建两个替代复制路径以使主复制路径总数达到三个，包括缺省主复制路径在内。

1. 在 pdb 和 RS2 之间创建一个名为 pdb_1 的替代主复制路径：
   a) 在 pdb 和 RS2 之间创建一个名为 pdb_1 的替代物理复制路径。
      在 PDS 中，输入：
      ```
      sp_replication_path "pdb", 'add', "pdb_1", "RS2", "RS2_user",
      "RS2_password"
      ```
   b) 从 RS2 到 pdb 创建一个名为 pdb_1 的相应替代主连接
      在 RS2 中，输入：
      ```
      create alternate connection to PDS.pdb
      named PDS.pdb_1
      set error class to rs_sqlserver_error_class
      set function string class to rs_sqlserver_function_class
      set username to pdb1_maint
      set password to pdb1_maint_ps
      with primary only
      ```
2. 在 pdb 和 RS1 之间创建另一个名为 pdb_2 的主复制路径：
   a) 在 pdb 和 RS1 之间创建一个名为 pdb_2 的替代物理复制路径。
      在 PDS 中，输入：
      ```
      sp_replication_path "pdb", 'add', "pdb_2", "RS1", "RS1_user",
      "RS1_password"
      ```
   b) 从 RS1 到 pdb 创建一个名为 pdb_2 的相应替代主连接
      在 RS1 中，输入：
      ```
      create alternate connection to PDS.pdb
      named PDS.pdb_2
      with primary only
      ```

删除 Replication Server 定义

将 **drop** 参数与 **sp_replication_path** 一起使用，以便从不是缺省主复制路径的物理复制路径中删除作为目标的 Replication Server。

无法删除缺省主复制路径；如果任何主复制路径具有绑定的对象，则无法删除该路径。

要删除作为目标的 RS1，请在 PDS 中输入：

```
sp_replication_path 'pdb', 'drop', "RS1"
```

创建逻辑主复制路径

可以将 **add** 和 **logical** 参数与 **sp_replication_path** 一起使用，以创建可用于将绑定到物理路径的数据和对象分配给多个 Replication Server 的逻辑主复制路径。

前提条件

创建相关物理主复制路径以支持逻辑主复制路径。

过程

如果将复制对象（如 dt1 维度表）绑定到 pdb_1，则 dt1 始终从 pdb_1 传送到 RS2。通过使用示例复制系统和三个物理主复制路径（缺省为 pdb_1 和 pdb_2），您可以创建名为 logical_1 的逻辑复制路径以将 dt1 通过 pdb_2 分配给 RS2。

注意： 无法将缺省路径添加到逻辑路径中。

1. 创建 logical_1 逻辑路径并将 pdb_1 添加为物理主复制路径：
   在 PDS 中，输入：
   ```
   sp_replication_path 'pdb', 'add', 'logical', 'logical_1', 'pdb_1'
   ```
   logical_1 仅通过 pdb_1 将数据发送到 RS1。
2. 将 pdb_2 添加为 logical_1 的物理主复制路径：
   在 PDS 中，输入：
   ```
   sp_replication_path 'pdb, 'add', 'logical', 'logical_1', 'pdb_2'
   ```
   logical_1 通过 pdb_1 将数据发送到 RS1，并通过 pdb_2 将数据发送到 RS2。

删除逻辑主复制路径中的元素

将 **drop** 和 **logical** 参数与 **sp_replication_path** 一起使用，以便从逻辑复制路径中删除元素。

在此示例中，logical_1 逻辑路径包含 pdb_1 和 pdb_2 物理路径，它们称为 logical_1 的元素。您可以从逻辑路径中删除元素。

警告！ 如果在现有的逻辑路径中删除或添加路径，则目标集可能会发生变化，复制的对象可能无法到达在发生变化之前能到达的目标。

1. 从 logical_1 中删除 pdb_1：
   ```
   sp_replication_path 'pdb', 'drop', 'logical', 'logical_1',
   'pdb_1'
   ```
   logical_1 逻辑路径现在仅包含 pdb_2 物理路径。绑定到 logical_1 的任何对象仅通过 pdb_2 进行复制。
2. 从 logical_1 中删除 pdb_2：
   ```
   sp_replication_path 'pdb', 'drop', 'logical', 'logical_1',
   'pdb_2'
   ```
   如果删除最后一个元素，并且没有绑定到逻辑路径的对象，Replication Server 将删除最后一个元素和整个逻辑路径，因为逻辑路径不能没有元素。

删除逻辑路径

将 **drop** 和 **logical** 参数与 **sp_replication_path** 一起使用以删除整个逻辑复制路径。

要删除整个逻辑路径，请不要在命令中指定 Replication Server 或元素。如果删除逻辑路径中的最后一个元素，Replication Server 将删除整个逻辑路径。

注意： 如果对象仍绑定到逻辑路径或物理路径，则无法删除这些路径。

要删除 logical_1 逻辑路径，请输入：

```
sp_replication_path 'pdb', 'drop', 'logical', 'logical_1'
```

设置分布模式

通过主 Adaptive Server 数据库中的多个主复制路径设置复制的分布模式。

前提条件

创建从主 Adaptive Server 到 Replication Server 的缺省和替代连接，并启用多线程 RepAgent。

过程

如果您将按对象绑定分布更改为按连接分布，RepAgent 将忽略所有对象绑定并显示警告。如果恢复为按对象绑定分布并重新启动 RepAgent，RepAgent 会保留绑定。

1. 设置分布模式：
   sp_config_rep_agent *database*, ‘multipath distribution model’, {‘connection' | 'object’}
   其中：
   - **multipath distribution model** – 是 **sp_config_rep_agent** 的分布模式参数
   - **connection** – 将模式设置为按连接分布
   - **object** – 将模式设置为按对象绑定分布（缺省值）
2. 停顿 Replication Server 并重新启动 RepAgent。
   请参见《Replication Server 管理指南第一卷》的“管理复制系统”的“Quiesce Replication Server”（停顿 Replication Server）中的“停顿复制系统”。

将对象绑定到复制路径

可以将 **bind** 参数与 **sp_replication_path** 一起使用，以将对象与物理或逻辑主复制路径相关联。在复制过程中，绑定对象始终遵循相同的路径。

要将对象绑定到主复制路径，请输入：

```
sp_replication_path dbname, 'bind', "object_type",
"[table_owner].object_name", "path_name"
```

其中：

- *object_type* – **table** 或 **sproc**（存储过程）。
- *[table_owner.]object_name* – 表或存储过程名称。

  注意： 如果不指定表所有者，则当对象是表时，绑定仅适用于数据库所有者 dbo 所有的表。
- *path_name* – 物理路径名或逻辑路径名。

例如，要将：

- t1 表绑定到 pdb_2 复制路径，请输入：

  ```
  sp_replication_path pdb, 'bind', "table", "t1", "pdb_2"
  ```

- 属于 owner1 的 t2 表绑定到 pdb_2 复制路径，请输入：

```
sp_replication_path pdb, 'bind', "table", "owner1.t2", "pdb_2"
```

- **sproc1** 存储过程绑定到 pdb_2 复制路径，请输入：

```
sp_replication_path pdb, 'bind', "sproc", "sproc1", "pdb_2"
```

- dt1 维度表对象绑定到 everywhere 逻辑路径，请输入：

```
sp_replication_path pdb, 'bind', "table", "dt1", "everywhere"
```

或者，可以在 *object_name* 中使用星号“*”或百分号“%”通配符或两者的组合来指定要绑定到某一路径的一定范围的名称或匹配字符。例如，要将名称与各种通配符组合相匹配的表绑定到 pdb_2 复制路径，请输入：

- `sp_replication_path pdb, 'bind', 'table', 'a*', "pdb_2"`
- `sp_replication_path pdb, 'bind', 'table', 'au%rs', "pdb_2"`
- `sp_replication_path pdb, 'bind', 'table', 'a*th%s', "pdb_2"`
- `sp_replication_path pdb, 'bind', 'table', 'authors%', "pdb_2"`

从复制路径中解除绑定对象

可以将 **unbind** 参数与 **sp_replication_path** 一起使用，以删除绑定的对象与物理或逻辑复制路径之间的关联。

要删除对象与一个或多个主复制路径之间的绑定，请输入：

```
sp_replication_path dbname, 'unbind', "object_type", "object_name",
{"path_name"|all}
```

其中：

- *object_type* – 指定对象类型，对象类型可以是 **path**、**table** 或 **sproc**（存储过程）。
- *[table_owner.]object_name* – 要解除绑定的表、存储过程或路径的名称。

  注意： 如果不指定表所有者，则当对象是表时，绑定仅适用于数据库所有者 dbo 所有的表。

- *path_name* | **all** – 指定物理或逻辑路径名或所有路径。如果指定 **path** 作为 *object_type*，则提供路径名作为 *object_name* 并指定 **all** 选项，Replication Agent 会从指定的路径名解除绑定所有对象。

例如：

- 要从 pdb_2 复制路径中删除绑定 t1 表，请输入：

```
sp_replication_path pdb, 'unbind', "table", "t1", "pdb_2"
```

- 要删除 t1 表上的所有绑定，请输入：

```
sp_replication_path pdb, 'unbind', "table", "t1", "all"
```

- 要删除 pdb_2 复制路径上的所有对象绑定，请输入：

```
sp_replication_path pdb, 'unbind', 'path', 'pdb_2', "all"
```

对象绑定以及 SQL 和 DDL 语句复制

可以通过未绑定到路径的任何对象的缺省路径或所有路径发送 SQL 和 DDL 语句。

在使用 SQL 语句和 DDL 复制时，如果将对象（如表）绑定到特定的复制路径，包含该对象的任何 SQL 或 DDL 语句将使用指定的复制路径。如果 SQL 或 DDL 语句包含未绑定到路径的任何对象，该语句将使用缺省复制路径或所有复制路径。

可以将 **ddl path for unbound objects** 与 **sp_config_rep_agent** 一起使用，以通过所有路径或缺省路径为出站对象发送 SQL 或 DDL 语句：

```
sp_config_rep_agent dbname, 'ddl path for unbound objects', {'all' |
'default'}
```

缺省设置为 **all**。

对象绑定和数据库重新同步

多路径复制通过所有可用的复制路径发送 **resync**、**resync purge** 和 **resync init** 数据库重新同步标记。

请参见《Replication Server 管理指南第二卷》的“复制系统恢复”的“Adaptive Server 的复制数据库重新同步”的“配置数据库重新同步”中的“向 Replication Server 发送 Resync Database 标记”。

对象绑定和 rs_ticket

多路径复制通过所有可用的复制路径发送执行 **rs_ticket** 的结果。必须过滤数据以获取与您相关的数据。

请参见《Replication Server 参考手册》的“RSSD 存储过程”中的“**rs_ticket**”。

在复制路径中更改配置值

将 **config** 参数与 **sp_replication_path** 一起使用以在替代复制路径中设置参数值。

您只能为替代复制路径更改口令和用户 ID。

要为替代路径更改参数值，请输入：

```
sp_replication_path dbname, 'config', "path_name",
 "config_parameter_name", "config_value"
```

其中 *config_parameter_name* 是 **rs_username** 或 **rs_password**。

例如：

- 要将 pdb_1 用于连接到 RS1 的用户名更改为“RS1_user”，请在 PDS 中输入：

  ```
  sp_replication_path pdb, 'config', "pdb_1", "rs_username",
  “RS1_user”
  ```

- 要将 pdb_1 用于连接到 RS1 的口令更改为“january，请在 PDS 中输入：

  ```
  sp_replication_path pdb, 'config', "pdb_1", "rs password",
  “january”
  ```

可以使用 **sp_config_rep_agent** 为缺省复制路径配置参数。请参见《Replication Server 参考手册》的“Adaptive Server 命令和系统过程”中的“**sp_config_rep_agent**”。

显示复制路径信息

可以在主数据库中将 **list** 参数与 **sp_replication_path** 一起使用以显示有关绑定和复制对象的信息。

```
sp_replication_path dbname, 'list', ['object_type'], ['object_name']
```

- *object_type* – 指定对象的类型：**path**、**table**、**sproc**（存储过程）。
- *object_name* – 显示特定对象的绑定关系。当想要指定对象的名称时，必须指定 *object_type*。

示例 1

要显示所有绑定的对象的路径关系，请不要指定 *object_type* 或 *object_name*:

```
sp_replication_path 'pdb','list'
go
```

您会看到:

```
Binding               Type     Path
--------------------- -------- --------------------
dbo.dt1               T        everywhere
dbo.sproc1            P        pdb_1
dbo.sproc1            P        pdb_2
dbo.t1                T        pdb_2
dbo.t2                T        pdb_1

(5 rows affected)

Logical Path                  Physical Path
----------------------------- --------------------
everywhere                    pdb_1
everywhere                    pdb_2

(2 rows affected)
Physical Path                   Destination
------------------------------- -------------------
pdb_1                           RS2
pdb_2                           RS1

(2 rows affected)
(return status = 0)
```

示例 2

要显示有关所有绑定的表的信息，请输入:

```
sp_replication_path 'pdb','list','table'
go
```

您会看到:

```
Binding               Type     Path
--------------------- -------- --------------------
dbo.dt1               T        everywhere
dbo.t1                T        pdb_2
dbo.t2                T        pdb_1
```

```
(3 rows affected)
(return status = 0)
```

示例 3

显示有关所有存储过程的信息：

```
sp_replication_path 'pdb','list','sproc'
go
```

您会看到：

```
Binding              Type     Path
-------------------- -------- --------------------
dbo.sproc1           P        pdb_2
dbo.sproc1           P        pdb_1
dbo.sproc2           P        pdb_1

(3 rows affected)
(return status = 0)
```

示例 4

要仅显示有关 **sproc1** 存储过程的信息，请输入：

```
sp_replication_path 'pdb','list','sproc','sproc1'
go
```

您会看到：

```
Binding              Type     Path
-------------------- -------- --------------------
dbo.sproc1           P        pdb_2
dbo.sproc1           P        pdb_1

(2 rows affected)
(return status = 0)
```

示例 5

要显示有关所有复制路径的信息，请输入：

```
sp_replication_path 'pdb','list','path'
go
```

您会看到：

```
Path                 Type     Binding
-------------------- -------- ----------------------
everywhere           T        dbo.dt1
pdb_1                P        dbo.sproc1
pdb_1                T        dbo.t2
pdb_2                P        dbo.sproc1
pdb_2                T        dbo.t1

(5 rows affected)
Logical Path                  Physical Path
----------------------------- ------------------------
everywhere                    pdb_1
everywhere                    pdb_2
```

```
(2 rows affected)
Physical Path                  Destination
------------------------------ ----------------------
pdb_1                          RS2
pdb_2                          RS1

(2 rows affected)
(return status = 0)
```

示例 6

要仅显示有关“everywhere”逻辑复制路径的信息，请输入：

```
sp_replication_path 'pdb','list','path','everywhere'
go
```

您会看到：

```
Path                  Type     Binding
--------------------- -------- -------------------------
everywhere            T        dbo.dt1

(1 rows affected)
Logical Path                   Physical Path
------------------------------ -------------------------
everywhere                     pdb_1
everywhere                     pdb_2

(2 rows affected)
Physical Path                  Destination
------------------------------ -------------------------
pdb_1                          RS2
pdb_2                          RS1

(2 rows affected)
(return status = 0)
```

注意： 您还会看到作为逻辑路径基础的物理路径。

示例 7

要仅显示有关 pdb_1 物理路径的信息，请输入：

```
sp_replication_path 'pdb','list','path','pdb_1'
go
```

您会看到：

```
Path                  Type     Binding
--------------------- -------- --------------------
pdb_1                 P        dbo.sproc1
pdb_1                 T        dbo.t2

(2 rows affected)
Physical Path                   Destination
------------------------------ --------------------
pdb_1                           RS2
```

```
(1 rows affected)
(return status = 0)
```

为 MSA 环境创建多个复制路径

可以使用复制定义和预订绑定复制数据库的复制连接和主数据库的主连接，以便在MSA环境中创建两个完整的复制路径。

1. 将事务分成两个组，并确保可以并行运行事务。
   例如，可以将事务分成两组对象，如表或存储过程。
2. 创建到主数据库的缺省主连接以及到复制数据库的缺省复制连接。
3. 创建到主数据库的替代主连接以及到复制数据库的替代复制连接。
4. 启用多线程 RepAgent 并为 RepAgent 启用两个复制路径，然后将对象绑定到复制路径。
5. 针对主数据库创建复制定义。
   如果缺省主连接和替代主连接位于不同 Replication Server 上，请在每个 Replication Server 上都创建复制定义。
6. 针对缺省主连接和缺省复制连接创建预订。
7. 针对替代主连接和替代复制连接创建预订。

用于热备份环境的多个复制路径

可以在热备份环境中使用替代连接和替代逻辑连接或建立端到端复制路径来提高复制性能。

在热备份环境中创建替代逻辑连接

可以使用 **create alternate logical connection** 在热备份环境中为现有的缺省逻辑连接创建替代逻辑连接。

可以使用不同的 Replication Server 来控制缺省逻辑连接和替代逻辑连接。活动数据库和备用数据库都必须支持具有多个 Replication Agent。在切换活动数据库和备用数据库时，请为每个逻辑连接（替代连接和缺省连接）执行 **switch active** 命令。在切换所有路径后，将完成热备份过程。

在管理热备份对的 Replication Server 上，输入：

```
create alternate logical connection to LDS.ldb
named conn_lds.conn_ldb
```

其中：

- *LDS* 和 *ldb* – 逻辑数据服务器和数据库名称。
- *conn_lds.conn_ldb* – 组成替代逻辑连接名称的数据服务器和数据库连接部分。

在热备份环境中创建替代连接

为替代逻辑连接创建到活动数据库的替代主连接或到备用数据库的替代复制连接。

在管理热备份对的 Replication Server 上，输入：

```
create alternate connection to ds_name.db_name
named conn_server.conn_db
...
[as {active|standby} for conn_lds.conn_ldb]
```

其中：

- *ds_name.db_name* – 数据服务器名称和活动或备用数据库名称。
- *conn_server.conn_db* – 替代活动或备用连接，由数据服务器名称和连接名称组成。*conn_ds* 必须与 *ds_name* 相同才能支持传入 Replication Agent 连接。
- *conn_lds.conn_ldb* – 组成替代逻辑连接名称的数据服务器和数据库连接部分。
- **active | standby** – 指定是否创建到活动数据库或备用数据库的替代连接。

为热备份环境创建多个复制路径

可以使用逻辑连接绑定备用数据库的复制连接和活动数据库的主连接，以便在热备份环境中的活动数据库和备用数据库之间创建两个完整的复制路径。

1. 将事务分成两个组，并确保可以并行运行两个组中的事务。
   例如，可以将事务分成两组对象，如表或存储过程。
2. 启用多线程 RepAgent 并为活动数据库和复制数据库的 RepAgent 启用两个复制路径，然后将对象绑定到复制路径。
3. 创建一个逻辑连接。请参见《Replication Server 参考手册》中的 **create logical connection**。
4. 使用 **rs_init** 将活动数据库和备用数据库添加到复制系统中。
5. 创建替代逻辑连接。
6. 为替代逻辑连接创建替代活动连接。
7. 使用 **admin who** 检查 REP AGENT 线程，并验证到热备份对的活动数据库的缺省连接和替代连接是否处于活动状态。
   例如，您会看到：

   ```
   31 REP AGEN   Awaiting Command     TOKYO_DS.pubs2
   ```
8. 为替代逻辑连接创建替代备用连接。

切换活动数据库和备用数据库

在从活动数据库切换到备用数据库时，您必须在具有多个复制路径的热备份环境中切换所有复制路径。

对于替代和缺省复制路径，切换过程是相同的。

1. 切换所有替代复制路径。
2. 切换缺省复制路径。

另请参见

- 切换活动数据库和备用 ASE 数据库 （第 76 页）

专用路由

专用路由仅分布特定主连接的事务。可以创建复制 Replication Server 的专用路由以复制高优先级事务或为特定主连接维护一个较不拥挤的路径。

共享路由位于主 Replication Server 和复制 Replication Server 之间，后者为所有源自主 Replication Server 的主连接分布事务。不要将共享路由绑定到特定连接。不绑定到专用路由的连接使用任何可用的有效共享路由。

只有在以下情况下才能创建专用路由：

- 存在从主 Replication Server 到目标 Replication Server 的共享路由，并且该共享路由是直接路由。如果 Replication Server 之间只有间接路由，则无法创建专用路由。
- 共享路由有效且未挂起。
- 共享路由的路由版本为 1570 或更高版本。

创建专用路由

使用 **create route** 和 **with primary at** 子句可创建专用路由。

例如，要在 `NY_DS.pdb1` 主连接的 RS_NY 主 Replication Server 和 RS_LON 复制 Replication Server 之间创建专用路由，请在 RS_NY 上输入：

```
create route to RS_LON
 with primary at NY_DS.pdb1
 go
```

在为特定连接创建专用路由后，从该连接到目标 Replication Server 的所有事务都将遵循该专用路由。

管理专用路由的命令

使用 **create route**、**drop route**、**resume route** 和 **suspend route** 可管理和监控专用路由。

在命令中包括 **with primary at** *dataserver.database* 子句可指定专用路由，其中 *dataserver.database* 是主连接名称。

请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**create route**、**drop route**、**suspend route** 和 **resume route**”。

命令	语法	命令和参数更改
create route	`create route to dest_replication_server { with primary at dataserver.database | set next site [to] thru_replication_server | [set username [to] user] [set password [to] passwd] [set route_param to 'value' [set route_param to 'value']...] [set security_param to 'value' [set security_param to 'value']...]}`	如果您在创建专用路由时指定用户 ID，则该用户 ID 必须是有效的用户。
drop route	`drop route to dest_replication_server [with primary at dataserver.database] [with nowait]`	必须先删除共享路由，然后才能删除专用路由。 在删除专用路由后，从指定的主连接到目标 Replication Server 的事务将通过共享路由。 **警告！** 仅在万不得已的情况下使用 **with nowait** 子句。 该子句会强制 Replication Server 删除路由，即使该路由在路由的外发队列中包含事务。因此，Replication Server 可能会放弃主连接中的一些事务。该子句指示 Replication Server 删除专用路由，即使该路由不能与目标 Replication Server 通信。 如果您使用该子句，请在以前的目标节点中使用 **sysadmin purge_route_at_replicate** 从目标系统表中删除预订和路由信息。 请参见《Replication Server 管理指南第一卷》的“管理路由”的“Drop Routes”（删除路由）中的“**drop route** 命令”。
suspend route	`suspend route to dest_replication_server [with primary at dataserver.database]`	

命令	语法	命令和参数更改
resume route	`resume route to dest_replication_server [with primary at dataserver.database] [skip transaction with large message]`	

显示专用路由信息

使用 **admin who** 可显示有关 Replication Server 之间的专用路由的信息。

在此示例中，有一个从 RS_NY 主 Replication Server 到 RS_LON 复制 Replication Server 的专用路由，用于 `NY_DS.pdb1` 主连接。在两个 Replication Server 中输入 **admin who**，您会看到：

- 在 `RS_LON` 上：

```
Spid Name       State               Info
45 SQT          Awaiting Wakeup     103:1  DIST NY_DS.pdb1
13 SQM          Awaiting Message    103:1 NY_DS.pdb1
32 REP AGENT    Awaiting Command    NY_DS.pdb1
16 RSI          Awaiting Wakeup     RS_LON
11 SQM          Awaiting Message    16777318:0 RS_LON
55 RSI          Awaiting Wakeup     RS_LON(103) /* Dedicated RSI
thread */
53 SQM          Awaiting Message    16777318:103 RS_LON(103) /
*Dedicated RSI outbound queue */
```

- 在 `RS_NY` 上：

```
Spid Name      State                Info
37 RSI USER   Awaiting Command    RS_NY(103) /*Dedicated RSI user
*/
32 RSI USER    Awaiting Command    RS_NY
```

请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**admin who**”。

用于多个复制路径的 Adaptive Server 监控表

在使用涉及用于 Adaptive Server 主数据库的 RepAgent 的多个路径进行复制期间，可以使用 Adaptive Server 监控表提供 Adaptive Server 状态的统计快照。可以通过这些表分析 Adaptive Server 性能。

表	说明
`monRepLogActivity`	从 Replication Agent 更新的监控计数器中提供信息
`monRepScanners`	提供 RepAgent 扫描程序任务的统计信息

表	说明
monRepScannersTotal-Time	提供有关 RepAgent 扫描程序任务将时间花在什么地方的信息
monRepSenders	提供有关 RepAgent 发送器任务的处理信息

如果您选择按连接分配，则可以使用 monRepSenders Adaptive Server 监控表中的其它字段提供数据分配的统计快照。

表 24. monRepSenders

字段	说明
NumberOfCommandsPro-cessed	每个 RepAgent 发送器线程为生成 LTL 而处理的命令数，如 **insert**、**delete**、**begin trans** 和 **commit trans**。
AvgBytesPerCmd	NumberOfBytesSent 与 NumberOfCommand-sProcessed 的比率。

请参见《Adaptive Server Enterprise 性能和调优系列：监控表》的“监控表简介”中的“Adaptive Server 中的监控表”。

替代主连接和复制连接的系统表支持

Replication Server 在 rs_databases 表中为每个主连接和复制连接创建一行，并包含 conn_id 列以唯一地标识特定行中的主连接或复制连接。

Replication Server 使用 dsname 和 dbname 列按连接名称标识替代连接，并按数据服务器和数据库名称标识缺省主连接或复制连接。dbid 标识连接所连接到的数据库的 ID。如果行用于缺省连接，则 connid 等于 dbid。如果行用于替代连接，则 connid 不等于 dbid。请参见《Replication Server 参考手册》中的“Replication Server 系统表”。

多处理器平台

您可以在对称多处理器 (SMP) 或单处理器平台上运行 Replication Server，因为 Replication Server 多线程体系结构支持这两种硬件配置。在多处理器平台上，Replication Server 线程以并行方式运行，因而可以提高性能和效率。

在单处理器平台上，Replication Server 线程以串行方式运行。

Replication Server 是一种 Open Server 应用程序。Replication Server 对多处理器的支持基于 Open Server 对多处理器的支持。这两种服务器都使用 POSIX 线程库（在 UNIX 平台上）和 WIN32 线程库（在 Windows 平台上）。有关 Open Server 多处理器计算机支持的详细信息，请参见《Open Server Server-Library/C 参考手册》。

当 Replication Server 处于单处理器模式时，全服务器范围的互斥锁强制串行线程执行。串行线程执行可保护全局数据、服务器代码和系统例程，以确保它们保持线程安全状态。

当 Replication Server 处于多处理器模式时，全服务器范围的互斥锁将被释放出来，各线程使用综合的线程管理技术来确保全局数据、服务器代码和系统例程保持安全状态。

启用多处理器支持

可以使用带 **smp_enable** 选项的 **configure replication server** 命令指定 Replication Server 是否利用多处理器计算机。

输入：

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

将 **smp_enable** 设置为 on 可指定多处理器支持；将 **smp_enable** 设置为 off 可指定单处理器支持。缺省值是 on。

smp_enable 是一个静态选项。在更改 **smp_enable** 的状态后，必须重新启动 Replication Server。

用于监控线程状态的命令

可以使用 **admin who** 命令或 **sp_help_rep_agent** 存储过程验证 Replication Server 线程状态。

- **admin who** – 提供有关所有 Replication Server 线程的信息
- **admin who_is_up** 或 **admin who_is_down** – 列出正在运行或未运行的 Replication Server 线程。
- **sp_help_rep_agent** – 提供有关 RepAgent 线程和 RepAgent User 线程的信息。

另请参见

- 验证和监控 Replication Server （第 3 页）

监控性能

Replication Server 提供了用于监控性能的监控器和计数器。

另请参见

- 使用计数器监控性能 （第 251 页）

分配队列段

您可以选择 Replication Server 为其分配稳定队列的段的磁盘分区。通过选择稳定队列的位置，您可以增强负载平衡和读/写分布。

Replication Server 将发往其它节点的消息存储在分区上。它将分区中的空间分配给稳定队列，并以称为段的 1MB 的块进行操作。每个稳定队列都包含要传送到另一个 Replication Server 或数据服务器的消息。在将数据发送到其目标之前，队列将一直保留这些数据。

rs_init 分配 Replication Server 的初始分区。您可能需要额外的分区，具体取决于 Replication Server 将消息发布到的数据库和远程 Replication Server 的数目。

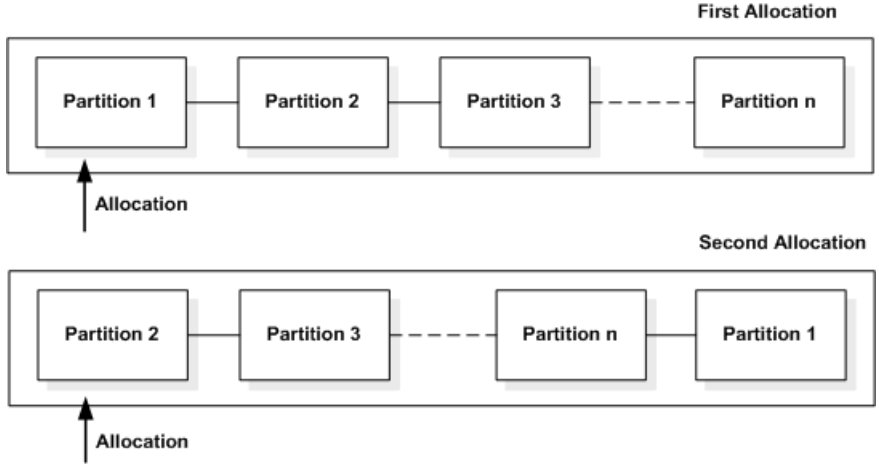
Replication Server 可以有任意多个大小不同的分区。各分区大小的总和是 Replication Server 用于排队事务的容量。

缺省分配机制

缺省情况下，Replication Server 将队列段指派给排序的分区列表中的第一个分区。

第一个分区充满时将变成最后一个分区，而下一个队列段将分配给新的第一个分区。在使用缺省方法时，段的滚动分配是自动的，用户无法控制。

图 22： 缺省分配机制



选择磁盘分配

若要选择段分配，请使用带 **set disk_affinity** 选项的 **alter connection** 或 **alter route** 命令。

语法为：

```
alter connection to dataserver.database
    set disk_affinity to [ 'partition' | 'off' ]
```

```
alter route to replication_server
    set disk_affinity to [ 'partition' | 'off' ]
```

partition 是分区的逻辑名，您将为其分配用于连接或路由的下一个段。

每个分配指令都称为“提示”，因为 Replication Server 可以覆盖分配（例如在分配的分区已满或已删除时）。如果 Replication Server 覆盖提示，它将根据缺省分配机制分配段。

Replication Server 每次分配新的队列段时，都会检查分配提示。每个提示都存储在 `rs_diskaffinity` 系统表中。每个分区可能有许多提示，但是每个稳定队列只能有一个提示。

是否可以成功使用磁盘分配提高性能，取决于节点的体系结构和其它特性。一种提高总体吞吐量的方法是将速度较快的设备与处理速度较慢的稳定队列关联。

此外，如果在所有连接到位后添加新的分区，则在填满现有分区之前，无法使用新的分区。您可以通过添加分配提示，迫使连接使用新的分区。

为稳定队列分配磁盘分区

您可以将不同的磁盘分区分配给不同的稳定队列。

例如，您可以使大小不同的分区可用于不同的数据库连接。在此示例中，我们将 10MB 和 20MB 的分区添加到 Replication Server，并为 TOKYO_DS 和 SEATTLE_DS 数据服务器指定了分配提示。

1. 使名为 `/dev/rds0a` 的设备上的分区 `P1` 和 `P2` 可用于 Replication Server。

   输入：

   ```
   create partition P1 on '/dev/rds0a' with size 20
   ```

   和

   ```
   create partition P2 on '/dev/rds0a' with size 10
   ```

2. 挂起至 TOKYO_DS 和 SEATTLE_DS 数据服务器的连接。

   输入：

   ```
   suspend connection to TOKYO_DS
   ```

   和

   ```
   suspend connection to SEATTLE_DS
   ```

3. 指定至 TOKYO_DS 和 SEATTLE_DS 数据服务器的连接的分配提示。

   输入：

   ```
   alter connection to TOKYO_DS.db1
   set disk_affinity to 'P1'
   ```

   和

```
alter connection to SEATTLE_DS.db5
set disk_affinity to 'P2'
```

4. 恢复至 TOKYO_DS 和 SEATTLE_DS 数据服务器的连接。

   输入：

   ```
   resume connection to TOKYO_DS
   ```

   和

   ```
   resume connection to SEATTLE_DS
   ```

删除提示和分区

可以使用带 **set disk_affinity to 'off'** 参数的 **alter connection** 或 **alter route** 命令删除分配提示。

例如：

```
alter connection to TOKYO_DS.db1
set disk_affinity to 'P1' to 'off'
```

此命令从 `rs_diskaffinity` 表中删除 `P1` 的分配提示。

您可以使用 **drop partition** 命令从 Replication Server 中删除分区。如果要删除的分区在 `rs_diskaffinity` 表中有一个或多个分配提示，则 Replication Server 将这些分配提示标记为删除，但是直到成功传送分区上存储的所有数据且删除该分区后，才会删除它们。

RMS 中的心跳功能

若要查看延迟信息，请使用命令行服务（即 Replication Monitoring Services (RMS)）中的心跳功能。

心跳功能使用存储过程 **rs_ticket** 来生成延迟信息，即将事务从主数据库移动到复制数据库所花费的时间量。RMS 将按照指定的时间间隔执行主数据库上的 **rs_ticket**。已生成的延迟信息将存储在复制数据库的某个表中。

RMS 提供了用于设置心跳过程和从复制数据库检索延迟信息的命令。只能通过 RMS 使用心跳功能。请参见《Replication Server 参考手册》的“Replication Monitoring Services API”中的“**get heartbeat**”和“**get heartbeat tickets**”。

使用计数器监控性能

Replication Server 提供了数百个不同的计数器，可用于在复制过程中监控各个时点和各方面的性能。

缺省情况下，只有在您选择激活计数器后，它们才会处于活动状态，但有几个计数器例外，它们始终处于活动状态。

要使用 RepAgent 计数器监控性能，请参见《Replication Server 管理指南第一卷》的“管理 RepAgent 和支持 Adaptive Server”中的“使用计数器监控 RepAgent 性能”。

用于查看计数器值的命令

可以使用一些命令随时查看当前计数器的值和其它性能信息。

可以使用：

- **admin stats** – 显示指定计数器的当前值。
- **admin stats, backlog** – 显示 Replication Server 稳定队列中的当前积压情况。
- **admin stats, { tps | cps | bps }** – 以每秒的事务数、命令数或字节数显示吞吐量。
- **admin stats, { md | mem | mem_in_use }** – 显示消息和内存信息。

还可以将计数器值保存（或刷新）到 RSSD。在 RSSD 中，可以使用标准 Transact-SQL 语句或 **rs_dump_stats** 存储过程计算和查看平均值和速率。

模块

在 Replication Server 中，模块是指可用来一起工作以执行特定服务的一组组件。

例如，稳定队列管理器 (SQM) 由一些可提供稳定队列服务的、逻辑上相关的组件组成。Replication Server 提供了各种计数器，可用于跟踪每个模块的每个实例的活动。

一些模块在 Replication Server 中只有一个实例。这些模块的实例单用模块名称就可标识。此类模块的示例包括：

- 系统表服务 (STS)
- 连接管理器 (CM)

其它模块可以在 Replication Server 中包含多个实例。若要对模块的每个实例进行唯一标识，您必须包括模块名称和实例 ID。示例包括：

- Replication Server 接口 (RSI)
- 分配器 (DIST)

- 数据服务器接口，调度程序线程 (DSI/S)

还有一些其它模块需要使用三个标识符来加以区分：模块名称、实例 ID 和实例值。示例包括：

- 稳定队列事务 (SQT) 线程
- 稳定队列管理器 (SQM)
- 数据服务器接口，执行程序线程 (DSIEXEC)

Replication Server 模块

Replication Server 具有一些常用的模块。

可以使用 Replication Server 命令直接对独立模块的计数器寻址。若要访问相关模块的计数器，请使用其父级模块的名称。

表 25. Replication Server 模块

模块名称	首字母缩略词	独立/相关
连接管理器	CM	独立
分配器	DIST	独立
数据服务器接口	DSI	独立
DSI 执行程序	DSIEXEC	与 DSI 相关
RepAgent 线程	REPAGENT	独立
Replication Server 接口	RSI	独立
RSI 用户	RSIUSER	独立
Replication Server 全局	SERV	独立
稳定队列管理器	SQM	独立
SQM Reader	SQMR	与 SQM 相关
SQM 事务管理器	SQT	独立
系统表服务	STS	独立
线程同步	SYNC	独立
同步元素	SYNCELE	与 SYNC 相关

计数器

每个计数器都包含一个说明性名称和一个显示名称，当您输入 RCL 命令和查看显示信息时可以使用二者来标识计数器。

若要查看有关 Replication Server 计数器的说明性信息和状态信息，请使用 **rs_helpcounter** 存储过程。

不同类别的计数器将提供不同类型的信息。虽然不是所有的计数器都可以划分到不同的类别中，但是在 Replication Server 显示计数器信息时将使用以下类别：

- 观测器 – 收集一段时间内事件的出现次数。例如，观测器可以收集从队列中读取某个消息的次数。Replication Server 将报告出现次数和每秒的出现次数。
- 监视器 – 收集在给定时间的度量情况。例如，监视器可以收集每个事务的操作次数。Replication Server 将报告观测次数、最后收集到的值、最大值和平均值。
- 计数器 – 收集各种度量值。测量持续时间的计数器与收集字节总数的计数器都包含在此组中。对于此类别，Replication Server 可以报告观测次数、总值、最后一个值、最大值、平均值和每秒的速率。

另请参见

- 查看有关计数器的信息 （第 261 页）

数据采样

您可以使用几个选项来收集数据。您可以选择是长时间或短时间（几秒）进行数据采样，还是仅进行一次数据采样。

您可以用以下两种方法中的任何一种来收集计数器统计信息：

- 通过使用 **display** 选项来执行 **admin stats**，可指示 Replication Server 收集指定时间段以及该时间段结束时的信息，以在计算机屏幕上显示收集到的信息。
- 通过使用 **save** 选项来执行 **admin stats**，可指示 Replication Server 收集指定时间段内指定观测次数的信息，并将该信息保存到 RSSD。

缺省情况下，在打开计数器之前，将不会通过相应的计数器来收集信息。当执行 **admin stats** 时，可以在特定时间段打开计数器。也可以通过设置 **stats_sampling** 配置参数以便在不定的时间段内打开采样功能。

打开样本收集功能将激活所有计数器。不过，可以仅针对所需的那些计数器或模块显示或保存统计信息。

计算机屏幕上显示的统计信息记录了在一次观测期间取得的事件数和计算值，例如平均值和比率。将统计信息发送到 RSSD 时，Replication Server 将保存多个连续观测周期的原始值，如观测值、总计、最后一个值和最大值。然后，您可以使用这些存储的值计算平均值和速率。

收集特定时间段内的统计信息

可以使用 **admin stats** 收集特定时间段内的统计信息。

admin stats 的语法是：

```
admin { stats | statistics } [, sysmon | "all"
        | module_name [, inbound | outbound ] [, display_name] ]
        [, server[, database ]       | instance_id ]
        [, display |, save [, obs_interval ] ]
        [, sample_period]
```

admin stats 允许您指定：

- 要对其进行采样的计数器
- 观测时间间隔长度和采样周期
- 将统计信息保存到 RSSD 还是在计算机屏幕上显示统计信息

注意： **admin stats** 还支持 **cancel** 选项。这将停止当前正在运行的命令。

缺省情况下，Replication Server 不会报告在采样周期内显示的观测次数为 0（零）的计数器。您可以使用 **configure replication server** 将 **stats_show_zero_counter** 配置设置为 on 来更改此行为。有关完整的语法和用法信息，请参见《Replication Server 参考手册》中的“Replication Server 命令”。

指定要对其进行采样的计数器

您可以指定所有计数器，也可以少至指定一个计数器的单个实例。

可以将以下参数与 **admin stats** 一起使用以指定计数器：

- **sysmon** – 对 Sybase 标记为对性能和调优最为重要的所有计数器进行采样。该值为缺省值。

  若要查看 **sysmon** 计数器的列表，请输入：

  ```
  rs_helpcounter sysmon
  ```
- **"all"** – 对所有计数器进行采样。
- **module_name** – 对特定模块的所有计数器进行采样。
- **module_name, display_name** – 对特定计数器的所有实例进行采样。使用 **sp_helpcounter** 可显示计数器的列表。
- **module_name, display_name, instance_id** – 对计数器的特定实例采样。若要查找某个实例的数字 ID，请执行 **admin_who** 并查看 `Info` 列。

  注意： 如果指定了实例 ID 并且模块是 SQT 或 SQM，则可以指定是否需要由计数器实例的入站或出站队列提供的信息。

例如，若要收集 **sysmon** 计数器在一秒内的统计信息，并将该信息发送到计算机屏幕，请输入：

```
admin stats, sysmon, display, 1
```

另请参见

- 模块 （第 251 页）

指定采样周期

可以用秒数来指定采样周期。

Replication Server 将收集指定的计数器在该秒数内的统计信息，并该信息其报告给屏幕或 RSSD。缺省值是 0（零）秒 – 这将导致所有计数器报告其当前值。

例如，若要收集所有计数器在一分钟内的统计信息，并在计算机屏幕上显示这些信息，请输入：

```
admin stats, "all", display, 60
```

指定统计信息的报告方式

可以将统计信息发送到计算机屏幕或 RSSD。

在计算机屏幕上显示统计信息

若要将统计信息发送到计算机屏幕上，请包含 **display** 选项。

在这种情况下，Replication Server 将在指定的时间段结束时执行单个观测。仅将观测到的统计信息发送到计算机屏幕。

例如，若要报告在五分钟的时间间隔内由所有读取器从所有队列读取到的阻止次数，请输入：

```
admin stats, sqm, blocksread, display, 300
```

当您使用 **display** 选项在非零时间段执行 **admin stats** 时，Replication Server 将执行以下操作：

1. 将所有计数器重置为 0。
2. 打开所有计数器。
3. 使您的会话在指定时间段内进入睡眠状态。
4. 关闭所有计数器。
5. 报告请求的数据。

在 RSSD 中保存统计信息

若要在 RSSD 中保存统计信息，请包含 **save** 选项，这将立即返回会话。

将统计信息发送到 RSSD 时，可以使用 *obs_interval* 指定在指定采样期间每个观察间隔的长度。*obs_interval* 可以是以秒为单位的数值，或带引号的时间格式字符串 hh:mm[:ss]。

例如，若要在 1 个小时 30 分钟之内按照 20 秒的时间间隔进行采样并将统计信息保存到 RSSD 中，请输入：

```
admin stats, "all", save, 20, "01:30:00"
```

若要在两分钟内按照 30 秒的时间间隔收集连接 108 的出站 SQT 的统计信息，请输入：

```
admin stats, sqt, outbound, 108, save, 30, 120
```

Replication Server 通过将采样周期除以观测时间间隔来确定观测时间间隔的数目。余数（以秒为单位）将添加到最后一个观测时间间隔（如果有）。

表 26. 采样周期和观测时间间隔

采样周期 (*sample_period*)	观测时间间隔 (*obs_interval*)	观测时间间隔的数目
60 秒	15	四个时长为 15 秒的时间间隔
75 秒	5	不允许 – 观测间隔必须 => 15 秒
60 秒	30	两个时长为 30 秒的时间间隔
130 秒	20	五个时长为 20 秒的时间间隔和最后的时长为 30 秒的时间间隔
10 秒	未指定	一个时长为 10 秒的时间间隔

当您使用 **save** 选项在非零时间段执行 **admin stats** 时，Replication Server 将启动后台线程来收集采样数据并立即返回会话。返回会话后，您可以使用 **admin stats, status** 命令来检查采样进度。后台线程将：

1. 截断 `rs_statrun` 和 `rs_statdetail` 系统表（如果配置参数 **stats_reset_rssd** 设置为 on）。
2. 重置所有计数器。
3. 打开所有计数器。
4. 在每个观测周期结束时将请求的计数器写入 RSSD。
5. 关闭所有计数器。

注意： 若要保留旧的采样数据，请将配置参数 **stats_reset_rssd** 设置为 off；或者，确保在转储 `rs_statrun` 和 `rs_statdetail` 中的任何所需信息之后，再使用 **save** 选项来执行 **admin stats**。可以使用 **rs_dump_stats** 过程转储这些表中的信息。

另请参见

- 使用 rs_dump_stats 过程 （第 260 页）

收集不定时间段内的统计信息

若要在不定周期内打开采样功能，请使用 **stats_sampling** 参数配置 Replication Server。

输入：

```
configure replication server
  set stats_sampling to "on"
```

Replication Server 将继续收集数据，直到您重新配置 Replication Server 以关闭采样。

```
configure replication server
   set stats_sampling to "off"
```

当您想要在计算机屏幕上查看数据或将收集到的数据发送到 RSSD 时，请使用 **admin stats**。

注意： 当 **stats_sampling** 设置为 on 时，使用 **admin stats** 时要小心。如果执行 **admin stats** 并指定一个非零时间段，则 Replication Server 将清除计数器，执行该命令并关闭 **stats_sampling**。

例如，若要收集两个连续的时长为 24 个小时的周期的统计信息，并将结果报告给计算机屏幕，则可以按照以下步骤操作：

第一天上午 8 点

1. 清除所有统计信息：
   输入：
   ```
   admin statistics, reset
   ```
2. 打开采样功能：
   输入：
   ```
   configure replication server
   set stats_sampling to "on"
   ```

第二天上午 8 点

1. 关闭采样功能，以确保在将统计信息转储到屏幕时 Replication Server 不会收集统计信息。
   输入：
   ```
   configure replication server
   set stats sampling to "off"
   ```
2. 将统计信息转储到屏幕上。
   输入：
   ```
   admin statistics, "all"
   ```
3. 清除所有统计信息：
   输入：
   ```
   admin statistics, reset
   ```
4. 打开采样功能：
   输入：
   ```
   configure replication server
   set stats_sampling to "on"
   ```

第三天上午 8 点

1. 关闭采样功能，以确保在将统计信息转储到屏幕时 Replication Server 不会收集统计信息。
   输入：
   ```
   configure replication server
   set stats sampling to "off"
   ```

2. 将统计信息转储到屏幕上。
   输入：
   ```
   admin statistics, "all"
   ```
3. 清除所有统计信息：
   输入：
   ```
   admin statistics, reset
   ```

在屏幕上查看统计信息

可以使用 **admin stats** 在计算机屏幕上显示单次采样的统计信息。

可以显示单个计数器实例、单个计数器、特定模块的所有计数器、通常最有用的或 **sysmon** 计数器或者所有计数器的统计信息。

在使用 **admin stats** 配置样本运行时，可以选择是否在屏幕上显示统计信息。

有关示例输出和完整语法以及用法信息，请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**admin stats**”。

另请参见

- 收集特定时间段内的统计信息 （第 254 页）

查看吞吐量速率

将 **admin stats** 和 **tps**、**cps** 或 **bps** 选项一起使用，可以按照每秒的事务数、命令数或字节数来查看当前吞吐量。

每秒的事务数
Replication Server 将根据在上一次重置计数器之后处理的事务数和经历的秒数来计算事务速率。将从若干模块获得此数据，其中包括 SQT、DIST 和 DSI 模块。

若要按照每秒的事务数查看吞吐量，请输入：

```
admin stats, tps
```

每秒的命令数
根据在上一次重置计数器之后处理的命令数和经历的秒数来计算每秒的命令数。将从 REPAGENT、RSIUSER、RSI、SQM、DIST 和 DSI 模块获得此数据。

若要按照每秒的命令数查看吞吐量，请输入：

```
admin stats, cps
```

每秒字节数
根据在上一次重置计数器之后处理的字节数和经历的秒数来计算每秒的字节数。将从 REPAGENT、RSIUSER、SQM、DSI 和 RSI 模块获得此数据。

若要按照每秒的字节数查看吞吐量，请输入：

```
admin stats, bps
```

查看有关消息和内存使用情况的统计信息

可以将 **admin stats** 和 **md** 选项一起使用以查看有关消息数的信息。将 **admin stats** 和 **mem** 或 **mem_in_use** 选项一起使用，可以查看有关内存使用情况的信息。

- 若要查看与分配器和 RSI 用户相关的消息传递的统计信息，请输入：

  ```
  admin stats, md
  ```
- 若要按照段大小查看当前的段使用情况，请输入：

  ```
  admin stats, mem
  ```
- 若要查看当前的内存使用情况（以字节为单位），请输入：

  ```
  admin stats, mem_in_use
  ```

查看稳定队列中的事务数

可以将 **admin stats** 与 **backlog** 选项一起使用，以查看入站和出站稳定队列中等待分配的事务数。

Replication Server 将按照段和块来报告数据，其中一个段等于 1MB，一个块等于 16K。将从 SQMRBacklogSeg 和 SQMRBacklogBlock 计数器获取此数据。

若要查看稳定队列积压，请输入：

```
admin stats, backlog
```

查看在 RSSD 中保存的统计信息

可以使用几个命令和过程查看 RSSD 中的统计信息。

发送到 RSSD 的统计信息存储在以下系统表中：

- `rs_statcounters` – 包含每个计数器的说明性信息
- `rs_statdetail` – 包含每个计数器每次采样时观测到的指标
- `rs_statrun` – 说明每次采样的情况

有关这些表的详细信息，请参见《Replication Server 参考手册》中的“Replication Server 系统表”。

您可以使用以下命令查看这些表中存储的统计信息：

- **select** 和其它 Transact-SQL 命令
- **rs_dump_stats**
- **rs_helpcounter** 可显示 `rs_statcounters` 中的信息

使用 rs_dump_stats 过程

rs_dump_stats 将 rs_statrun 和 rs_statdetail 系统表格中的内容转储为一个 CSV 文件，可以将此 CSV 文件装载到电子表格中以供分析。

有关完整的语法和用法信息，请参见《Replication Server 参考手册》的“RSSD 存储过程”中的“**rs_dump_stats**”。

若要使用 **rs_dump_stats**，请登录到 RSSD 并执行存储过程。例如：

```
1> rs_dump_stats
2> go
```

rs_dump_stats 的示例输出

注意： 输出右侧包含一些注释，用于对此示例进行解释。这些注释不是 **rs_dump_stats** 输出的一部分。

```
Comment:  Sample of rs_dump_stats output
Nov  5 2005 12:29:18:930AM                          *Start time stamp*
Nov  5 2005 12:46:51:350AM                          *End time stamp*
16                                                  *No of observation
intervals*
1                                                    *No of min between
                                                      observations*
16384                                             *SQM bytes per block*
64                                              *SQM blocks per segment*
CM                                                  *Module name*
13                                                  *Instance ID*
-1                                                  *Instance value*
dCM                                                 *Module name*
CM: Outbound database connection request            *Counter external
name*
CMOBDBReq                                         *Counter display name*
13003          , , 13,  -1                        *Counter ID, instance
ID,
                                                        instance value*
ENDOFDATA                                          *EOD for counter*

CM: Outbound non-database connection requests       *Counter external
name*
CMOBNonDBReq                                      *Counter display name*
13004          , , 13,  -1                        *Counter ID, instance
ID,
                                                        instance value*
Nov  5 2005 12:29:18:930AM,  103,  103,  1,  1   *Dump ts, obs,
total,
                                                         last, max*
Nov  5 2005 12:30:28:746AM,  103,  103,  1,  1
Nov  5 2005 12:31:38:816AM,  107,  107,  1,  1
Nov  5 2005 12:32:49:416AM,  104,  104,  1,  1
Nov  5 2005 12:33:58:766AM,  114,  114,  1,  1
...
Nov  5 2005 12:46:51:350AM,  107,  107,  1,  1
```

```
ENDOFDATA                                              *EOD for counter*

CM: Outbound 'free' matching connections found   *Counter external
name*
CMOBFreeMtchFound                                 *Counter display name*
13005          , , 13,  -1                        *Counter ID, instance
ID,
                                                          instance value*

Nov  5 2005 12:29:18:930AM,  103,  103,  1,  1   *Dump ts, obs,
total,
                                                      last, max*

Nov  5 2005 12:30:28:746AM,  103,  103,  1,  1
...
Nov  5 2005 12:46:51:350AM,  2,  2,  1,  1
ENDOFDATA                                           *EOD for counter*
```

查看有关计数器的信息

可以使用 **rs_helpcounter** 系统过程，查看有关 `rs_statcounters` 表中存储的计数器的说明性信息。

- 要查看包含计数器的模块列表和 **rs_helpcounter** 过程的语法，请输入：

  ```
  rs_helpcounter
  ```

- 要查看有关指定模块的所有计数器的说明性信息，请输入：

  ```
  rs_helpcounter module_name[, short | long ]
  ```

  如果您输入 **short**，Replication Server 将会输出每个计数器的显示名称、模块名称和计数器说明。

  如果您输入 **long**，Replication Server 将输出每个计数器在 `rs_statcounters` 中对应的每个列。

  如果您不输入第二个参数，Replication Server 就会输出每个计数器的显示名称、模块名称和外部名称。

- 要列出与某个关键字匹配的所有计数器，请输入：

  ```
  rs_helpcounter keyword [, short |, long ]
  ```

- 要列出具有指定状态的计数器，可以使用以下语法：

  ```
  rs_helpcounter { sysmon | internal | must_sample
          | no_reset | old | configure }
  ```

有关详细的语法和用法信息，请参见《Replication Server 参考手册》的“RSSD 存储过程”中的“**rs_helpcounter**”。

重置计数器

可以使用 **admin stats, reset** 命令将所有计数器（从不重置的计数器除外）重置为 0（零）。

输入：

```
admin stats, reset
```

如果尚未使用 **stats_sampling** 参数启用采样，则计数器的值为零。在非零采样周期运行 **admin stats**，会将计数器设为零，然后打开采样，并在完成采样后关闭计数器采样，最后将计数器重置为零。如果采用周期为零，则将报告当前计数器值。

如果已启用采样，则在使用 **admin stats** 时要小心。使用 **stats_sampling** 配置启用采样之后，将累计计数器值。若发出 **admin stats** 并指定采样周期，将使 Replication Server 在采样运行之后清除所有计数器并禁用采样（将 **stats_sampling** 设置为 off）。

生成性能报告

可以使用 **rs_stat_populate** 和 **rs_stat_genreport** 存储过程生成性能报告。

在升级到 Replication Server 15.1 之后，必须将此脚本装载到 RSSD 中：

```
$SYBASE/$SYBASE_REP/scripts/
rs_install_statreport_v1510_[ase|asa].sql
```

在加载脚本之后，运行 **rs_stat_populate** 和 **rs_stat_genreport** 以生成以下性能报告：

- Replication Server 性能概述 – 有关 Replication Server 的概述信息，例如 DIST 处理、DSI 处理等。
- Replication Server 性能分析 – 基于关键 Replication Server 计数器的性能分析和调优建议。脚本文件中提供了详细说明。
- 活动对象标识结果 – 列出活动表和过程名称、所有者名称、执行次数等。

有关 **rs_stat_populate** 和 **rs_stat_genreport** 的详细信息，请参见脚本文件，其中包含语法、示例和其它信息。

错误和例外处理

了解 Replication Server 的各种处理错误方法。

有关解决特定错误的信息，请参见《Replication Server 故障排除指南》。

常规错误处理

在可以访问数据服务器和其它 Replication Server 时，Replication Server 会将消息传递给它们，而当连接断开时，Replication Server 会对消息进行排队。可以使用 Sybase Central 监控复制系统状态，并在出现故障时进行故障排除。

通常，网络和数据服务器出现短期故障时并不需要进行特别的错误处理或干预。故障排除之后，复制系统组件会自动恢复其工作。对于持续时间较长的故障，如果没有足够的磁盘空间来对消息进行排队，或者，如果必须重新配置复制系统才能避免故障，则可能会需要进行干预。

某些系统组件（例如，Replication Server 分区或主数据库）的故障也需要用户干预复制系统恢复过程。

Replication Server 对错误如何响应取决于错误的类型、错误的来源以及 Replication Server 的配置方式。Replication Server：

- 在它的错误日志文件中记录错误。
- 根据配置设置响应数据服务器错误。
- 如果事务在数据库中无法提交，将事务写入例外日志，以便手工解决。
- 系统重新启动后，检测重复的事务。

另请参见

- 复制系统恢复 （第 283 页）

错误日志文件

了解复制系统中排除 Replication Server 和 RepAgent 故障时可访问的错误日志文件。

要查看写入系统表的跳过的事务，您可以访问 Adaptive Server，以查看管理某个指定的数据库的 Replication Server。请参见《Replication Server 故障排除指南》。

Replication Server 允许进行用户可定义的错误处理来响应数据服务器错误。

另请参见

- 数据服务器错误处理 （第 267 页）

Replication Server 错误日志

Replication Server 错误日志是一个文本文件，Replication Server 在该文件中写入信息性消息和错误消息。

缺省情况下，Replication Server 错误日志的文件名为 `repserver.log`，位于您在其中启动 Replication Server 的目录中。当您启动 Replication Server 时，或在 Replication Server 的运行文件中时，您可以使用 **-E** 命令行标志指定错误日志文件的名称和位置

Replication Server 错误日志中的消息类型

Replication Server 错误日志中包含几种消息类型。每条日志消息都以一个字母开头，用于表示消息类型。

表 27. Replication Server 错误日志中的消息类型

错误代码	说明
I	信息性消息。
W	对于尚未引发错误但可能需要注意的情况的警告。例如，资源用尽。
E	不会防碍继续进行处理的错误，例如，节点不可用。
H	某个 Replication Server 线程已停止工作。例如，网络连接丢失。
F	致命错误。导致 Replication Server 退出的严重错误。例如，使用不正确的配置启动 Replication Server。
N	内部错误。这些错误是由 Replication Server 软件中的异常情况导致的。请向 Sybase 技术支持部门报告这些错误。

信息性消息

Replication Server 错误日志中的信息性类型。

错误日志中的信息性消息的格式如下：

```
I. date: message
```

消息开头的字母“I”意味着该消息是信息性消息。它不表示有错误发生。例如，当 Replication Server 删除预订时，会输出以下消息：

```
I. 95/11/01 05:41:54. REPLICATE RS: Dropping
subscription authors_sub for replication definition
authors with replicate at <SYDNEY_DS.pubs2>
```

```
I. 95/11/01 05:42:02. SQM starting: 104:-2147483527
authors.authors_sub
```

```
I. 95/11/01 05:42:12. SQM Stopping: 104:-2147483527
authors.authors_sub
```

```
I. 95/11/01 05:42:20. REPLICATE RS: Dropped
subscription authors_sub for replication definition
authors with replicate at <SYDNEY_DS.pubs2>
```

错误和警告消息

Replication Server 错误日志中的错误和警告消息。

信息性消息以外的其它消息的格式如下：

```
severity, date. ERROR #error_number thread_name(context) -
source_file(line) message
```

如果消息是警告，则格式中的“ERROR”将变为“WARNING”。

参数为：

- *severity* – W、E、H、F 或 N，与 Replication Server 错误日志中的消息类型相对应。
- *date* – 发生错误的日期和时间。
- *error_number* – Replication Server 错误号。
- *thread_name* – 出现错误的 Replication Server 线程的名称。有关 Replication Server 线程的详细信息，请参见《Replication Server 管理指南第一卷》中的“Replication Server 技术概述”。
- *context* – 提供一些在发生错误时的线程上下文的相关信息。
- *source_file* – Replication Server 源代码中报告错误的程序文件。
- *line* – Replication Server 源代码中报告错误的程序文件中的行号。
- *RS_language* – 指定 Replication Server 消息的语言。
- *message* – Replication Server 中的消息的完整文本（使用 *RS_language* 配置参数中指定的语言显示）。有些消息还包括数据服务器中的消息，或 Replication Server 使用的某个组件库中的消息。

注意： Replication Server 在未提供具体信息的消息中添加问号 (?)。例如，如果在初始化过程中出现错误，而 Replication Server 可能还未完成某些内部结构，它就会输出问号来代替尚未收集的信息。

下面是数据服务器的 Replication Server 错误日志条目：

```
E. 95/11/01 05:30:52. ERROR #1028 DSI(SYDNEY_DS.pubs2)
- dsiqmint.c(3522)Message from server:
 Message: 2812, State: 4, Severity: 16 --
 ’ Stored procedure ’ upd_authors’ not found.
```

```
H. 95/11/01 05:30:53. THREAD FATAL ERROR #5049
DSI(SYDNEY_DS.pubs2) - dsiqmint.c(3529)
The DSI thread for database ’ SYDNEY_DS.pubs2’ is being
shutdown because of error action mapped from data server
error ’ 2812’. The error was caused by output command ’ 1’
mapped from source command ’ 2’ of the transaction.
```

这些消息表示 Adaptive Server 返回了错误号 2812，从而导致 Replication Server 采取了 **stop_replication** 操作。您可以为数据服务器错误指定其它操作。

查找 Replication Server 错误日志的名称

可以使用 **admin log_name** 命令查找当前 Replication Server 错误日志文件的名称。

Replication Server 会显示该日志文件的路径，如下面的 UNIX 示例所示：

```
Log File Name
 ----------------------------
 /work/sybase/SYDNEY_RS/SYDNEY_RS.log
```

更改为新的 Replication Server 日志文件

可以使用 **admin set_log_name** 命令开始新的错误日志文件。

此命令会关闭当前日志文件并打开一个新的日志文件。后续消息将会写入到该新日志文件中。

例如，在 UNIX 中输入：

```
admin set_log_name, '/work/sybase/SYDNEY_RS/951101.log'
```

如果 Replication Server 未能创建并打开新的日志文件，上一个日志仍会保持活动状态。

RepAgent 错误日志消息

所有 RepAgent 错误、跟踪和信息消息都会记录到 Adaptive Server 错误日志文件中。

每条消息都用字符串“RepAgent (*dbid*)”标识记录错误的 RepAgent，该字符串出现在消息的第一行。*dbid* 是记录错误的 RepAgent 的数据库标识号。

以下是信息性消息：

```
RepAgent(dbid): Recovery of transaction log is
complete. Please load the next transaction log dump and
then start up the Rep Agent Thread with
sp_start_rep_agent, with 'recovery' specified.
```

Adaptive Server 错误日志是一个文本文件。这些消息以 Adaptive Server 上指定的语言输出。RepAgent 会记录从 Adaptive Server 事务日志传输已复制的对象并将它们转换为命令时发生的错误和信息性消息。RepAgent 错误的范围通常是 9200 到 9299。

Adaptive Server 会根据错误的严重性和可恢复性执行操作。有些错误只是提供信息，而其它一些错误则会导致 Adaptive Server 重试引起错误的操作，直至成功，还有一些其它错误表示错误太严重，无法继续，而且 RepAgent 会关闭。请参见《Adaptive Server Enterprise 故障排除：错误消息高级分析》。

示例 RepAgent 错误消息

常见的 RepAgent 错误消息和可能的解决方案。

- 在本例中，Replication Server 上没有提供 RepAgent 登录名：

  ```
  RepAgent(6): Failed to connect to Replication
  Server. Please check the Replication Server,
  username, and password specified to
  ```

```
sp_config_rep_agent. RepSvr = repserver_name, user =
RepAgent_username
```

```
RepAgent(6): This Rep Agent Thread is aborting due
to an unrecoverable communications or Replication
Server error.
```

您必须将 RepAgent 的登录名添加到 Replication Server 或更改 RepAgent 的登录名。

- 在本例中，RepAgent 无法连接到 Replication Server：

```
RepAgent(7): The Rep Agent Thread will retry the
connection to the Replication Server every 60
second(s). (RepSvr = repserver_name.)
```

检查 Replication Server 的状态。如果 Replication Server 已关闭，请解决问题，然后重新启动。否则，请等待解决可能存在的网络问题。

数据服务器错误处理

Replication Server 允许对数据服务器错误进行用户可定义的错误处理。将相应的错误类分配给指定的连接并自定义分配的错误类。错误操作应与数据服务器返回的错误相匹配。

用于错误处理的 RCL 命令和系统过程

可以使用一些 RCL 命令和 Adaptive Server 系统过程管理错误和错误类。

表 28. 用于错误处理的 RCL 命令和系统过程

命令	说明
assign action	为一个或多个数据服务器或 Replication Server 错误指定错误处理操作
alter error class	更改现有的错误类
create error class	创建新的错误类
drop error class	删除现有的错误类
alter connection	将一个错误类与现有的数据库连接相关联
create connection	将一个错误类与新的数据库连接相关联
rs_helpclass	Adaptive Server 存储过程，它显示每个现有错误类、函数字符串类及其主 Replication Server 的名称；对于继承类，还会显示父类的名称。
rs_helperror	Adaptive Server 存储过程，它显示映射到给定数据服务器或 Replication Server 错误号的 Replication Server 错误操作。

缺省错误类

Replication Server 提供 **rs_sqlserver_error_class** 作为缺省 Adaptive Server 错误类，提供 **rs_repserver_error_class** 作为缺省 Replication Server 错误类，并为非 ASE 数据库提供缺省错误类。不能修改这些缺省的错误类。

表 29. 非 ASE 错误类

数据库	类名
IBM DB2	**rs_db2_error_class**
IBM UDB	**rs_udb_error_class**
Microsoft SQL Server	**rs_msss_error_class**
Oracle	**rs_oracle_error_class**
Sybase IQ	**rs_iq_error_class**

另请参见

- 为错误类指定主节点 （第 269 页）

非 ASE 数据库的本机错误代码

当 Replication Server 建立到非 ASE 复制服务器的连接时，Replication Server 会验证是否对连接启用从非 ASE 复制服务器返回本机错误代码的选项。

如果未启用此选项，Replication Server 会记录警告消息，表明连接有效但错误操作映射可能不正确。

请参见“Replication Server Options”的《面向访问服务的 Enterprise Connect Data Access Option for ODBC 用户指南》的“配置访问服务库”的“配置属性类别”的“目标交互属性”中的“**ReturnNativeError**”以在 Enterprise Connect™ Data Access (ECDA) Option for ODBC 中为复制服务器设置此选项。

创建错误类

可以使用 **create error class** 创建您自己的错误类。

错误类是用于对错误操作分配进行分组的名称。**create error class** 将错误操作从模板错误类复制到新错误类。

您可以定义一个错误类，以用于由同一类型的数据服务器管理的所有数据库。例如，您可以将缺省 Adaptive Server 错误类 `rs_sqlserver_error_class` 用于任何 Adaptive Server 数据库。无需创建另一个错误类，除非数据库有特殊的错误处理要求。

注意： 在使用连接配置文件创建连接时，连接配置文件将指派错误类。连接配置文件将为特定数据服务器预定义错误类。请参见《Replication Server 管理指南第一卷》

的“管理数据库连接”的“准备要进行复制的数据库”的“准备要进行复制的非 ASE 服务器”中的“连接配置文件”。

要创建错误类，请输入：

```
create [replication server] error class error_class
[set template to template_error_class]
```

replication server 选项指定要创建 Replication Server 错误类。您可以使用 **set template to** 选项并将另一个错误类作为模板来创建错误类。

示例

以下示例创建一个名为 **pubs2_error_class** 的错误类，并且不使用模板错误类：

```
create error class pubs2_error_class
```

此示例根据缺省 Replication Server 错误类 **my_rs_err_class** 创建 Replication Server 错误类 **rs_repserver_error_class**：

```
create replication server error class my_rs_err_class
set template to rs_repserver_error_class
```

此示例将 **rs_oracle_error_class** 作为模板创建适用于 Oracle 数据库的 **my_error_class** 错误类：

```
create error class my_error_class
set template to rs_oracle_error_class
```

为错误类指定主节点

您必须先指定一个主节点，然后才能修改缺省错误类。

最初，`rs_sqlserver_error_class` 和其它缺省非 ASE 错误类没有主节点。由于您只能在主节点上为类创建服务器范围的错误类，因此，请使用 **create error class** 将某个 Replication Server 指定为错误类的主节点。

例如，对于 Adaptive Server，在主节点上执行 **create error class rs_sqlserver_error_class**。请确保所有其它 Replication Server 具有源自主节点的直接或间接路由。

另请参见

- 更改错误类的主 Replication Server （第 271 页）

指定错误操作

您可以为数据服务器返回的错误指定不同的错误操作。

数据服务器返回的所有错误的缺省错误操作是 **stop_replication**。

这也是最为严重的操作：它会挂起数据库的复制，就好像您输入了 **suspend connection** 命令。要将不太严重的操作指派给要分别处理的错误，请使用 **assign action** 命令。

另请参见

- 为数据服务器错误指定操作 （第 272 页）

更改错误类

alter error class 将模板错误类中的错误操作复制到要更改的错误类中并覆盖具有相同错误代码的错误操作。

要更改错误类，请输入：

```
alter [replication server] error class error_class
set template to template_error_class
```

replication server 选项指定要更改 Replication Server 错误类。

例如，要将 rs_sqlserver_error_class 作为模板更改用于 Oracle 数据库的 my_error_class，请输入：

```
alter error class my_error_class
set template to rs_sqlserver_error_class
```

初始化新的错误类

在创建了新的错误类后，您可以使用错误类（例如，系统提供的 rs_sqlserver_error_class）的错误操作来初始化新的错误类。

为此，请使用 **rs_init_erroractions** 存储过程：

```
rs_init_erroractions new_error_class, template_class
```

例如，要根据模板错误类 rs_sqlserver_error_class 初始化错误类 pubs2_error_class，请输入：

```
rs_init_erroractions pubs2_error_class, rs_sqlserver_error_class
```

然后，使用 **assign action** 命令来更改各个错误的操作。

删除错误类

drop error class 命令删除一个错误类以及与其关联的所有操作。

当您删除某个错误类时，该错误类一定不能正用于某个活动的数据库连接。**drop error class** 的语法是：

```
drop [replication server] error class error_class
```

例如，要删除 pubs2_error_class 错误类，请输入：

```
drop error class pubs2_error_class
```

您无法删除 rs_sqlserver_error_class 或任何缺省非 ASE 错误类。

另请参见

- 缺省错误类 （第 268 页）

更改错误类的主 Replication Server

可以使用 **move primary** 命令更改错误类的主节点。

当您要将主节点从一个 Replication Server 更改到另一个 Replication Server 以便可以通过新路由分配错误操作时，则必须更改错误类的主节点。例如，如果您要从复制系统删除当前作为错误类主节点的 Replication Server，则必须使用此命令。

执行 **move primary** 之前，请确认存在以下路由：

- 从新的主节点到每个将使用该错误类的 Replication Server
- 从当前主节点到新的主节点
- 从新主节点到当前主节点

用于错误类的 **move primary** 命令的语法是：

```
move primary
   of [replication server] error class class_name
   to replication_server
```

在要指定为错误类新主节点的 Replication Server 上执行 **move primary** 命令。

参数为：

- **replication server** – 此选项指定您要更改 Replication Server 错误类的主 Replication Server。如果要修改数据服务器错误类，请保留此选项。
- *class_ name* – 要更改其主 Replication Server 的错误类的名称。
- *replication_server* – 为错误类指定新的主 Replication Server。

以下命令将 `pubs2_error_class` 错误类的主节点更改为 TOKYO_RS Replication Server（在此处输入命令）：

```
move primary of error class pubs2_error_class
 to TOKYO_RS
```

对于缺省错误类 `rs_sqlserver_error_class`，除非您指定一个 Replication Server 作为主节点，否则没有 Replication Server 会成为主节点。您必须先指定主节点，然后才能使用 **assign action** 命令更改缺省错误操作。

要为缺省错误类指定主节点，请在该 Replication Server 中执行以下命令：

```
create error class rs_sqlserver_error_class
```

执行了此命令后，您就可以使用 **move primary** 命令来更改错误类的主节点了。

显示错误类信息

可以使用 **rs_helpclass** 存储过程显示主 Replication Server 和复制系统中现有的错误类和函数字符串类的名称。

例如：

```
rs_helpclass error_class
```

```
Error Class(es)              PRS for class
 --------------               ---------
rs_sqlserver_error_class    Not Yet Defined
```

请参见《Replication Server 参考手册》的“RSSD 存储过程”中的“**rs_helpclass**”。

为数据服务器错误指定操作

可以使用 **assign action** 命令指定对数据服务器可返回到 Replication Server 的错误采取的操作。

```
assign action
  {ignore | warn | retry_log | log | retry_stop | stop_replication}
  for error_class
  to server_error1 [, server_error2]...
```

您必须先在主节点上创建缺省错误类，然后才能使用 **assign action** 更改缺省错误操作。*data_server_error* 参数是数据服务器错误号。

可以使用 **create connection** 和 **alter connection**，这复制数据库上的特定连接指定错误类。

在创建错误类的 Replication Server 上输入六个可能的错误操作之一：**ignore** 是严重性最低的操作，而 **stop_replication** 是严重性最高的操作。有关错误号、错误消息、相应的缺省错误操作和说明，请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**assign action**”。

当某个事务导致多个错误时，Replication Server 只选择一个操作，即指派给发生的任何错误的最严格的操作。要将错误返回到缺省错误操作 **stop_replication**，您必须明确地重新指派它。

还可以指定 Replication Server 如何响应在 SQL 语句复制期间可能发生的 SQLDML 行数错误。如果发生 SQLDML 行数错误，则说明 SQL 语句复制后在主数据库和复制数据库中更改的行数不匹配。Replication Server 的缺省错误操作是停止复制。Replication Server 的缺省错误类是 `rs_repserver_error_class`。

下面是行计数错误消息示例：

```
DSI_SQLDML_ROW_COUNT_INVALID 5186
Row count mismatch for the SQL Statement Replication
command executed on 'mydataserver.mydatabase'. The
command impacted 10 rows but it should impact 15 rows.
```

指定错误操作示例

例如，若要指示 Replication Server 忽略 Adaptive Server 错误 5701 和 5703，请使用以下命令：

```
assign action ignore
 for rs_sqlserver_error_class
 to 5701, 5703
```

例如，要在 Replication Server 遇到行计数错误（由错误号 5186 指示）时发出警告，请输入：

```
assign action warn
   for rs_repserver_error_class to 5186
```

另请参见

- 数据服务器的错误操作 （第 273 页）
- 缺省错误类 （第 268 页）

数据服务器的错误操作

可以为数据服务器错误指定一些错误操作。

表 30. 数据服务器错误的 Replication Server 操作

操作	说明
ignore	假设命令成功执行并且没有要处理的错误或警告情况。此操作可以用于表示成功执行的返回状态。
warn	记录一条警告消息，但不会回退该事务或中断执行。
retry_log	回退该事务并重试。尝试重试的次数是用 **configure connection** 命令来设置的。如果重试后该错误仍然存在，会将该事务写入例外日志，然后继续执行下一个事务。
log	回退当前事务并将其记录在例外日志中，然后继续执行下一个事务。
retry_stop	回退该事务并重试。尝试重试的次数是用 **configure connection** 命令来设置的。如果重试后再次发生错误，将挂起该数据库的复制。
stop_replication	回退当前事务并挂起数据库的复制。这相当于使用 **suspend connection** 命令。此操作为缺省设置。 由于此操作会停止数据库的所有复制活动，所以要找出不用关闭数据库连接就能处理的数据服务器错误，然后将它们分配给其它操作，这一点很重要。

显示为错误号指定的操作

可以使用 **rs_helperror** 存储过程显示为错误号指定的操作。

rs_helperror 的语法如下：

```
rs_helperror server_error_number [, v]
```

其中 *server_error_number* 参数是您需要其信息的错误的数据服务器错误号。**v** 参数指定“详细”报告。当您提供此选项时，**rs_helperror** 还会显示 Adaptive Server 错误消息文本（如果有）。请参见《Replication Server 参考手册》的“RSSD 存储过程”中的“**rs_helperror**”。

行计数验证

Replication Server 缺省启用行计数验证，并自动显示错误消息和执行缺省错误操作以纠正不同的行计数验证错误，例如，行计数不匹配。您可以配置 Replication Server 错误类以启用不同的错误操作。

连接将自身与两种错误类类型（数据服务器错误类和 Replication Server 错误类）相关联。必须首先将 Replication Server 错误类与连接相关联，然后 Replication Server 才能查询 Replication Server 错误类是否存在缺省 Replication Server 错误操作的覆盖操作。只能将一个连接与一个 Replication Server 错误类相关联。然而，可以将一个 Replication Server 错误类与多个连接相关联。可以使用带 **set replication server error class** 参数的 **create connection** 和 **alter connection** 命令，将 Replication Server 错误类与连接相关联。

Replication Server 响应错误时，它首先查找指派给连接的 Replication Server 错误类。如果 Replication Server 找不到 Replication Server 错误类，则 Replication Server 使用分配给服务器的缺省 **rs_repserver_error_class** 错误类。

注意： 对于自定义函数字符串中的那些命令，Replication Server 会忽略行计数验证。

控制行计数验证

可以使用 **dsi_row_count_validation** 禁用行计数验证。

如果表行不同步并希望绕过缺省错误操作和消息，则可以将 **dsi_row_count_validation** 设置为 **off** 以禁用行计数验证。

缺省情况下，**dsi_row_count_validation** 设置为 **on** 以启用行计数验证。

可以使用 **configure replication server** 在服务器级设置 **dsi_row_count_validation** 以影响所有复制数据库连接，或使用 **alter connection** 为指定的数据库和数据服务器的连接设置该参数。例如：

- 要为所有数据库连接禁用行计数验证，请输入：

  ```
  configure replication server
  set dsi_row_count_validation to 'off'
  ```

  在使用 **dsi_row_count_validation** 执行 **configure replication server** 后，必须挂起并恢复到 Replication Server 的所有数据库连接。设置中的更改在您重新开始数据库连接后生效。
- 要为特定连接启用行计数验证（SYDNEY_DS 数据服务器中的 `pubs2` 数据库），请输入：

  ```
  alter connection to SYDNEY_DS.pubs2
  set dsi_row_count_validation to 'on'
  ```

  在为连接设置 **dsi_row_count_validation** 时，无需挂起并重新开始数据库连接；该参数会立即生效。不过，新设置将影响 Replication Server 在您执行命令后处理的

一批复制对象。更改此设置不会影响 Replication Server 当前正在处理的一批复制对象。

在行计数验证错误消息中显示表名

行计数验证错误消息显示表名。

如果您正在使用：

- 连续模式的日志顺序逐行复制 – Replication Server 记录并显示表名、表所有者名称和标识引发事务失败的输出命令的数字。Replication Server 仅记录表名的前 30 字节。您可以启用 DSI_CHECK_ROW_COUNT_FULL_NAME 跟踪来将显示的表名的最大长度扩展为 255 字节。
- 高容量自适应复制 (HVAR) 或实时装载 (RTL) – Replication Server 记录并显示 HVAR 和 RTL 编译产生的内部 **join-update** 和 **join-delete** 语句。您无法获取引发失败的事务的特定命令，因为 HVAR 或 RTL 已将该命令编译为 HVAR 和 RTL 处理的一部分。Replication Server 可显示的 **join-update** 和 **join-delete** 语句的最大长度是 128 字节，包括“`...\0`”尾随字符串。

此示例包括：

- 主节点 – 包含名为 `ThisTableHasANameLongerThan30Characters` 的表的 `pdb1` 主数据库，该表具有三列和三行。

id	name	age
1	John	40
2	Paul	38
3	George	37

- 复制节点 – 包含具有相同名称 `ThisTableHasANameLongerThan30Characters` 的表的 `rdb1` 主数据库，该表具有两行，`id` 列的值为 1 和 3。

如果针对 `pdb1` 执行以下命令：

```
update ThisTableHasANameLongerThan30Characters set age = 20
```

则对于每种类型的复制模式，错误消息的显示有所不同。在以下模式中：

- 连续模式的日志顺序逐行复制：

```
I. 2010/06/07 01:30:21.DSI received Replication Server
error #5185 which is mapped to WARN by error action mapping.
W. 2010/06/07 01:30:21.WARNING #5185 DSI EXEC(103(1)
ost_replnx6_61.rdb1) - /dsiexec.c(11941)
Row count mismatch for the command executed on
'ost_replnx6_61.rdb1'.The command impacted 0 rows but it
should impact 1 rows.
```

```
I. 2010/06/07 01:30:21.The error was caused by output
command #3 of the failed transaction on table
'dbo.ThisTableHasANameLongerThan30C'.
```

注意：表名将截断为缺省 30 字节。

如果您打开 **DSI_CHECK_ROW_COUNT_FULL_NAME** 跟踪以允许使用错误消息可显示的最大表名长度 255 字节，则错误消息的最后一行显示完整的表名：

```
I. 2010/06/07 02:22:55.The error was caused by output
command #3 of the failed transaction on table
'dbo.ThisTableHasANameLongerThan30Characters'.
```

- HVAR 或 RTL 复制：

```
W. 2010/06/07 02:06:56.WARNING #5185 DSI EXEC(103(1)
ost_replnx6_61.rdb1) - i/hqexec.c(4047)
Row count mismatch for the command executed on
'ost_replnx6_61.rdb1'.The command impacted 1 rows but it
should impact 2 rows.
I. 2010/06/07 02:06:56.(HQ Error):update
ThisTableHasANameLongerThan30Characters set age = w.age
from ThisTableHasANameLongerThan30Characters
t,#rs_uThisTab...
I. 2010/06/07 02:06:57.The DSI thread for database
'ost_replnx6_61.rdb1' is shutdown.
```

例外处理

如果 Replication Server 提交的事务失败，Replication Server 将在 RSSD 的例外日志中记录该事务。值班的复制系统管理员必须解决例外日志中的事务。

由于出现了诸如重复键、列值检查和磁盘空间不足这样的错误，事务可能就会失败。而权限不足、版本控制冲突和对象引用无效等原因，也可能会导致事务被拒绝。

因为跳过某个事务会造成不一致，并且可能会对系统有负面影响，所以您应该定期检查例外日志中记录的任何事务，并解决处理它们。对事务的最佳解决方法可能会取决于该事务源自的客户端应用程序。例如，如果失败的事务对应于现实生活中的事件，例如，提取现金，则必须以某种方式来应用该事务。

有关跳过事务会造成的后果的详细信息，请参见《Replication Server 故障排除指南》。

另请参见

- 访问例外日志 （第 278 页）

处理失败的事务

了解处理需要手工干预的失败事务时建议采用的过程。

挂起数据库连接

当数据服务器因临时故障（例如，数据库或日志文件中缺少空间）开始拒绝事务时，您可以在改正错误之前挂起数据库连接。

如果没有挂起数据库连接，Replication Server 会将事务写入到数据库的例外日志中。由于必须手工解决处理这些事务，所以您可以在错误情况得到改正之前先关闭连接，以节省时间。

当挂起数据库连接时，Replication Server 会将事务存储在稳定的队列中。当连接恢复时，存储的事务将被发送到数据服务器。

要停止从 Replication Server 到数据库的事务流，请使用 **suspend connection**：

```
suspend connection to data_server.database
```

该命令需要 **sa** 权限，并且必须在管理数据库的 Replication Server 上输入。

分析和解决问题

确定事务失败的原因、进行改正或做出调整，然后重新提交事务。

1. 从例外日志中检索事务的列表。
2. 研究事务，以确定失败的原因和最佳解决方法。
3. 根据您的计划，对事务进行解决处理。例如，您可以改正权限问题，然后重新提交事务。
4. 从例外日志中删除已解决处理过的事务。

例如，如果事务失败的原因是维护用户没有足够的权限，那么，请授予该维护用户所需的权限，然后重试该事务。

另请参见

- 访问例外日志 （第 278 页）
- 从例外日志中删除事务 （第 280 页）

恢复连接

可以使用 **resume connection** 重新启动已挂起的数据库连接的事务流。

无论是使用 **suspend connection** 命令有意挂起的连接，还是因错误操作而被 Replication Server 挂起的连接，都使用同一个命令。

```
resume connection to data_server.database
[skip transaction]
```

该命令需要 **sa** 权限，并且必须在管理数据库的 Replication Server 上输入。

使用 **skip transaction** 子句可以指示 Replication Server 忽略队列中的第一个事务。如果每次恢复连接时事务总是失败，您可能会需要这样做。

访问例外日志

Replication Manager 提供一个图形界面以查看和管理例外日志中的事务。

显示例外日志中的事务

可以使用 **rs_helpexception** 存储过程显示例外日志中所有事务的摘要。

```
rs_helpexception [transaction_id, [, v]]
```

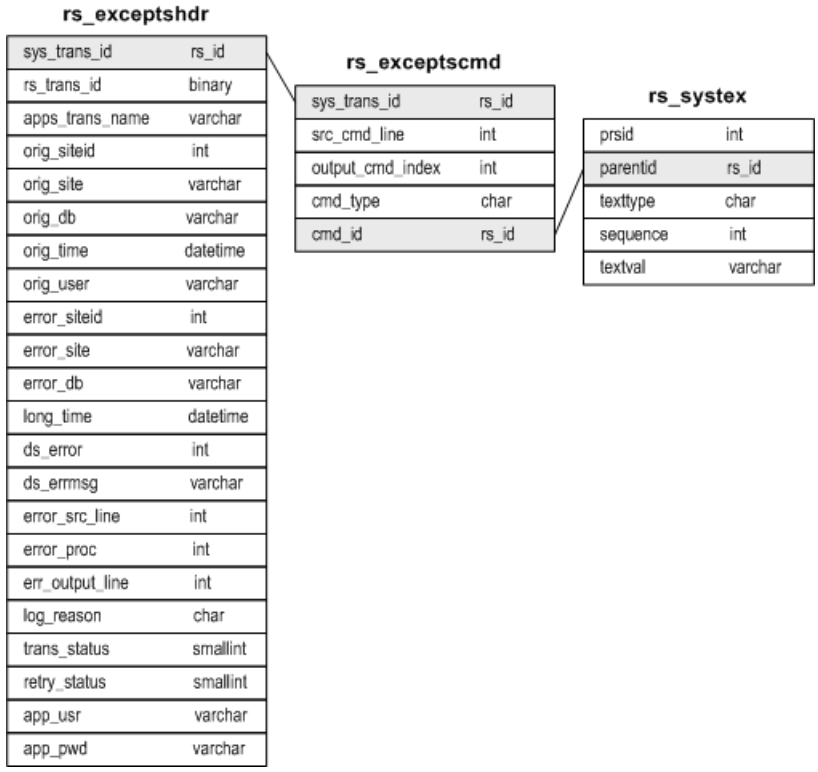
如果提供了有效的 *transaction_id* 和 **v**（后者表示“详细”报告），**rs_helpexception** 将显示详细的事务说明。使用不带参数的 **rs_helpexception** 可以获取例外日志中所有事务的 *transaction_id* 号。

查询例外日志系统表

可以在 `sys_trans_id` 列上连接 `rs_exceptshdr` 和 `rs_exceptscmd` 系统表以检索有关例外的信息。

您还可以连接 `rs_exceptscmd` 和 `rs_systext` 系统表以检索事务的文本。为此，请将 `rs_exceptscmd` 中的 `cmd_id` 列连接到 `rs_systext` 中的 `parentid` 列。

图 23： 例外日志系统表



`rs_exceptshdr` 系统表包含例外日志中有关事务的说明性信息，包括以下内容：

- 由用户指派的事务名称
- 事务起始的节点和数据库
- 在源节点上提交事务的用户
- 导致事务要被记录在例外日志中的错误的有关信息

例如，要检索给定数据库的例外事务列表，请使用以下查询：

```
select * from rs_exceptshdr
 where error_site = 'data_server'
 and error_db = 'database'
 order by log_time
```

要使用给定系统事务 ID 检索事务的源和输出文本，请使用：

```
select t.texttype, t.sequence,
 t.textval
 from rs_systext t, rs_exceptscmd e
 where e.sys_trans_id = sys_trans_id
and t.parentid = e.cmd_id
 order by e.src_cmd_line, e.output_cmd_index,
 t.sequence
```

请参见《Replication Server 参考手册》中的“Replication Server 系统表”以查看这些 Replication Server 系统表中的所有列的列表。

从例外日志中删除事务

可以使用存储过程从 RSSD 例外日志中删除事务。

- **rs_delexception** – 删除例外日志中的单个事务。语法为：

  ```
  rs_delexception [transaction_id]
  ```

  可以使用不带参数的 **rs_delexception** 显示例外日志中的事务摘要。如果您提供了有效的 *transaction_id*，**rs_delexception** 会删除事务。使用不带参数的 **rs_helpexception** 或 **rs_delexception** 即可查找事务的 *transaction_id*。
  例如，要删除 ID 号为 1234 的事务，请输入：

  ```
  rs_delexception 1234
  ```

- **rs_delexception_id** – 删除按事务 ID 指定的一组事务。语法为：

  ```
  rs_delexception_id transaction_id_start [,transaction_id_end]
  ```

  例如，要删除 ID 号在 1234 到 9800 之间的所有事务（含这两个 ID 号），请输入：

  ```
  rs_delexception_id 1234, 9800
  ```

- **rs_delexception_date** – 删除按事务日期指定的一组事务。语法为：

  ```
  rs_delexception_date transaction_date_start
  [,transaction_date_end]
  ```

  例如，要从例外日志中删除起始日期在 2010 年 10 月 1 日和 2010 年 10 月 31 日之间（含这两个日期）的所有事务，请输入：

  ```
  rs_delexception_date "10/01/2010", "10/31/2010"
  ```

- **rs_delexception_range** – 删除按发起节点或用户或目标节点指定的一组事务。语法为：

  ```
  rs_delexception_range
  {{"origin"|"org"}, "origin_data_server.origin_database" |
  , {"destination"|"dest"},
  "destination_data_server.destination_database" |
  , "user", "origin_user"}
  ```

  例如，要从例外日志中删除源自 SYDNEY_DS 数据服务器的 `south_db` 数据库的事务，请输入：

  ```
  rs_delexception_range "org", "SYDNEY_DS.south_db"
  ```

有关完整的用法信息和更多示例，请参见《Replication Server 参考手册》的“RSSD 存储过程”中的存储过程说明。

请参见《Replication Server 管理指南第一卷》的“使用 Sybase Central 管理复制环境”的“设置复制环境”的“管理 Replication Server 对象”的“队列”中的“查看队列数据”。

DSI 重复检测

DSI 会记录最后提交或写入例外日志的事务，所以它可以在系统重新启动之后检测是否有重复。每个事务都由一个唯一的源数据库 ID 和一个随每个事务递增的源队列 ID 来标识。

通过执行为数据服务器的函数字符串类定义的函数字符串，可以将从每个源数据库提交的最后一个事务记录在该数据服务器上。对于系统定义的类，可以通过 **commit** 命令的函数字符串（即 **rs_commit** 函数）完成此操作。每个函数字符串类均支持 **rs_get_lastcommit** 函数，此函数会为每个源数据库返回 `origin_qid` 和 `secondary_qid`。`secondary_qid` 是用于实现预订或取消实现预订的查询的 ID。

从每个源数据库写入例外日志的最后一个事务的 `origin_qid` 和 `secondary_qid` 均会被记录到 `rs_exceptslast` 系统表中。但是，由 **sysadmin log_first_tran** 命令显式记录到日志中的事务不会记录在这个系统表中。这些事务会被记录到日志中，但不会被跳过。

当启动或重新启动 DSI 时，它会获得 **rs_get_lastcommit** 函数返回的以及 `rs_exceptslast` 系统表中存储的 `origin_qid`。它会假定队列中 `origin_qid` 小于这两个值中较大值的任何事务为重复事务，并会忽略该事务。

如果错误地修改了数据服务器或 `rs_exceptslast` 系统表中存储的 `origin_qid` 值，则可能会忽略非重复的事务，或者可能会再次应用重复的事务。如果您怀疑系统中发生这种情况，请检查存储的值，并将它们与数据库的稳定队列中的事务进行比较，以确定值的有效性。如果值是错误的，则必须直接修改它们。

有关如何转储队列中的事务的详细信息，请参见《Replication Server 故障排除指南》。

系统事务的重复检测

了解如何检测和解决系统事务执行故障。

虽然可以将 **truncate table** 和某些支持的 DDL 命令复制到备用数据库和复制数据库中，但不会记录这些命令。有关每个 DDL 命令的信息，请参见《Adaptive Server Enterprise 参考手册》。

Replication Server 会将这些命令作为系统事务来复制。进行复制时，Replication Server 将 **truncate table** 或类似命令“夹”在两个完整事务之间。第一个事务的执行会被记录在复制数据库 `rs_lastcommit` 表的 `secondary_qid` 列和 `origin_qid` 列中。如果 Replication Server 记录第二个事务，则在完成系统事务时，Replication Server 会清除 `secondary_qid` 列。

在系统发生故障后，以下消息表示系统命令尚未完成，而连接已关闭。

```
5152 DSI_SYSTRAN_SHUTDOWN,"There is a system
transaction whose state is not known. DSI will be
shutdown."
```

您必须验证系统事务中的命令是否已在复制数据库上执行。

- 如果该命令已执行，或者，如果您自己执行该命令，您可以在恢复连接后跳过队列中的第一个事务，继续执行第二个事务。在复制 Replication Server 上，输入：

  ```
  resume connection to data_server.database
  skip transaction
  ```
- 如果该命令尚未执行，您可以修复问题，然后执行队列中的第一个命令。在复制 Replication Server 上，输入：

  ```
  resume connection to data_server.database
  execute transaction
  ```

您必须在使用 **resume connection** 时包含 **skip transaction** 或 **execute transaction** 子句。否则，Replication Server 将无法正确地重置 `secondary_qid`，并且会再次出现错误消息。

另请参见

- 支持的 DDL 命令和系统过程 （第 55 页）

复制系统恢复

虽然 Replication Server 可以承受大多数故障情况，而且能够自动从这些故障状态中恢复，但有些故障却需要用户干预。了解如何找出这些故障以及用于恢复的过程，设计这些过程的目的是：通过恢复丢失的和损坏的数据并将这些数据恢复到其原先的状态，来维护复制系统的完整性。

在设计、安装和管理您的复制系统时，您应当考虑如何进行备份和恢复。我们假定：转储会定期执行，而且已经准备好了适当的工具并进行了相应设置，以便着手进行恢复。

在有关恢复的讨论中，“当前”Replication Server 指的是要恢复的数据库（例如，RSSD）所在的 Replication Server。“上游”Replication Server 具有指向当前 Replication Server 的直接或间接路由。“下游”Replication Server 是一个当前 Replication Server 具有指向它的直接或间接路由的 Replication Server。

例如，如果主数据库和复制数据库之间存在复制延迟而导致无法单独使用复制恢复数据库，则可以在复制环境中重新同步复制数据库。

另请参见

- 创建协调的转储 （第 290 页）
- Adaptive Server 的复制数据库重新同步 （第 321 页）

如何使用恢复过程

如果您要使用恢复过程，在执行恢复步骤时，每次都要写下或者核对这些步骤。这些信息可以帮助 Sybase 技术支持部门确定您在恢复过程中进行到了哪一步（如有必要）。

每种故障情况具有相应的故障症状和恢复过程。

警告！ 恢复过程只适用于特定于该过程的故障情况。请不要使用这些恢复过程来解决复制系统问题（例如无法复制数据）。尝试使用恢复过程来处理那些非指定的情况可能会使您的问题变复杂，使恢复操作更麻烦。

有关诊断和解决问题的帮助，请参见《Replication Server 故障排除指南》。

另请参见

- 从分区丢失或分区故障中恢复 （第 290 页）
- 从截断的主数据库日志中恢复 （第 294 页）
- 从主数据库故障中恢复 （第 296 页）
- 从 RSSD 故障中恢复 （第 298 页）
- Adaptive Server 的复制数据库重新同步 （第 321 页）

配置复制系统以支持 Sybase 故障切换

了解 Replication Server 版本 12.0 和更高版本如何支持 Adaptive Server Enterprise 版本 12.0 中提供的 Sybase 故障切换功能。

Sybase 故障切换使您能够将两个 12.0 版或更高版本的 Adaptive Server 配置为协同服务器。如果主协同 Adaptive Server 发生故障，该服务器的设备、数据库和连接可以由协同 Adaptive Server 来接管。

您可以按非对称或对称方式来配置高可用性系统。

非对称配置 包括两个 Adaptive Server，它们虽然在物理上分别位于不同的计算机，但是共享相同的系统设备、系统/主数据库、用户数据库和用户登录。这两台服务器互相连接，这样，如果其中一台服务器发生故障，则另一台可承担其工作量。协同 Adaptive Server 用作“热备份”，在发生故障切换之前不执行任何工作。

对称配置 也包括两个运行在不同计算机上的 Adaptive Server，但每个 Adaptive Server 都具有其自己的系统设备、系统/主数据库、用户数据库和用户登录，可以完全独立运行。如果发生故障切换，任何一个 Adaptive Server 均可作为另一个 Adaptive Server 的协同服务器。

无论是这两种设置中的哪一种，两台计算机都配置为双向存取，这样，两台服务器就都能够看到和访问磁盘。

在复制系统中，Replication Server 与 Adaptive Server 建立了许多连接，您可以启用或禁用 Replication Server 建立的到 Adaptive Server 的数据库连接的故障切换支持。当您启用了故障切换支持时，连接到发生故障的 Adaptive Server 的 Replication Server 将会自动切换到协同计算机，以重新建立网络连接。

请参见“Adaptive Server Enterprise”的《在高可用性系统中使用 Sybase 故障切换》。

另请参见

- Sun Cluster 2.2 的高可用性 （第 345 页）

为 Replication Server 启用故障切换支持

您应当为系统中的每个 Replication Server 启用故障切换支持：为 RSSD 连接启用一次，为所有其它从指定的 Replication Server 到 Adaptive Server 的数据库连接启用一次。

对 RSSD 连接以外的其它连接，您不能为单个连接启用故障切换支持。

对于所有从 Replication Server 到 Adaptive Server 的连接，Replication Server 中故障切换支持的缺省设置为“关闭”。

要进行不间断复制，您应该为所有连接启用故障切换支持。不过，在某些情况下，当协同服务器的工作量超过其承载能力时，您可能需要禁用连接的故障切换。

在 Replication Server 中如何实现 Sybase 故障切换

要配置从 Replication Server 到 Adaptive Server 的 Sybase 故障切换，必须将 Adaptive Server 配置为允许连接故障切换。

当 Adaptive Server 处于故障切换协同模式时，如果主协同服务器发生故障，辅助协同服务器会接管其工作量。那些需要更新到 RSSD 的未完成事务或操作将会失败。Replication Server 会重试现有连接，但新的连接将进行故障切换。

对于“数据服务器接口”(DSI) 连接，DSI 会在短暂休眠后重试失败的连接。

对于 RSSD 连接，在故障切换期间所执行的用户命令将会失败。但内部操作（例如，对定位符和磁盘段的更新）应该会成功。DSI 应该包括 RSSD 对象的复制。

如果异步命令（例如，预订、路由和备份命令）已被接受，但尚未执行完毕，则可能会被拒绝，或遇到错误而需要恢复。例如，创建预订命令可能已被接受，但预订可能仍在创建中。

注意： 故障切换支持不能替代热备份。虽然热备份会保留数据库的副本，但故障切换支持从不同的计算机访问同一个数据库。对于从 Replication Server 到热备份数据库的连接，故障切换支持的工作方式是相同的。

Sybase 故障切换支持要求

要使用故障切换支持，应满足一些要求。

- 要启用故障切换支持，Replication Server 必须连接到已配置了故障切换功能的 12.0 版或更高版本的 Adaptive Server。
- “Replication Server 系统数据库”(RSSD) 和用户数据库的故障切换是直接通过 Adaptive Server 配置的。
- 故障切换支持只响应 Adaptive Server 的故障切换；也就是说，Replication Server 的故障切换并不受支持。
- Adaptive Server 负责 RepAgent 线程故障切换以及故障切换/故障恢复后将其重新连接到 Replication Server。
- 每个 Replication Server 都配置其自己的连接。

为 RSSD 连接启用故障切换支持

在安装 Replication Server 后，请编辑配置文件以便为 RSSD 连接启用故障切换支持。

在安装新的 Replication Server 时，也可以使用 **rs_init** 启用故障切换支持。请参见《Replication Server 配置指南》中的“使用 rs_init 配置 Replication Server 和添加数据库”。

1. 使用文本编辑器打开 Replication Server 的配置文件。

   缺省文件名为 Replication Server 的名称加上“`.cfg`”扩展名。配置文件中每个条目包含一行。
2. 找到 `RSSD_ha_failover=no` 行并将其更改为 `RSSD_ha_failover=yes`。

可通过设置 RSSD_ha_failover=no，为 RSSD 连接禁用故障切换支持。

这些更改将会立即生效；也就是说，您不必重新启动 Replication Server 就可以启用故障切换支持。

为非 RSSD 数据库连接启用故障切换支持

可以使用带 **ha_failover** 的 **configure replication server**，为从 Replication Server 到 Adaptive Server 的新数据库连接启用故障切换支持。

请参见“Adaptive Server Enterprise”的《在高可用性系统中使用 Sybase 故障切换》。

1. 必要时启动 Replication Server。

   请参见《Replication Server 管理指南第一卷》的“管理复制系统”中的“启动 Replication Server”。
2. 登录到 Replication Server：

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

   其中，*user_name* 必须具有“管理员”权限。用 **-S** 标志指定 Replication Server 的名称。

   当您的登录名被接受时，**isql** 会显示一个提示：

   ```
   1>
   ```
3. 将 **ha_failover** 设置为 on。

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

配置复制系统以防止数据丢失

了解一些建议的措施，用于在发生无法挽救的数据库错误时防止数据丢失。如果使用得当，通过这些措施，您可以使用系统恢复过程来恢复复制数据。

恢复的保存间隔

Replication Server 能够存储从消息源发送来的消息，然后它可将这些消息转发到它们的目标。要想增大重建稳定队列后恢复联机消息的可能性，您可以为 Replication Server 之间的路由设置保存间隔（以分钟为单位）。

保存间隔是消息转发后存储的时长。您也可以为 Replication Server 的物理或逻辑数据库连接设置保存间隔，从而允许 Replication Server 将消息保存在 DSI 出站队列中。

要查找某个路由或连接的当前保存间隔，请使用 **admin who, sqm** 命令。Save_Int:Seg 列存放着两个值。冒号之前的值是保存间隔。冒号之后的值是稳定队列中第一个保存的段。

Replication Server 之间的路由

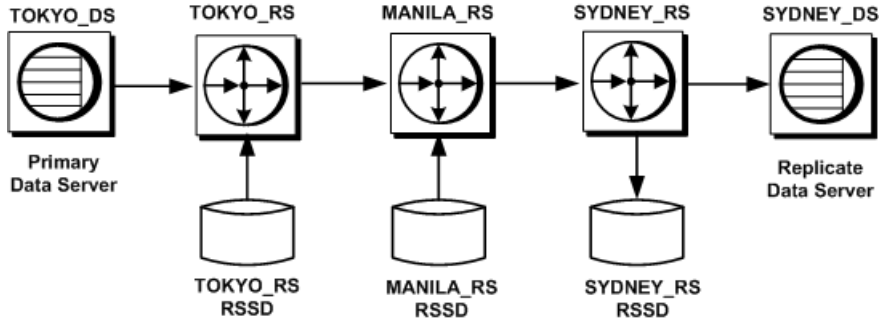
您可以为 Replication Server 之间的路由设置保存间隔以进行消息恢复。

如果 Replication Server 已挂起路由，或者，网络或数据服务器连接已断开，积压的消息可能就会堆积在 Replication Server 的稳定队列中。恢复这些消息的可能性将会随时间的推移而减小。源 Replication Server 可能已经将消息从其稳定队列中删除，而且数据库日志可能已经被截断。

如果您为 Replication Server 之间的每个路由都设置了 *save_interval* 参数，则您允许每个 Replication Server 在路由的下一个节点确认已接收到消息后还至少将消息保留由该值指定的一段时间。这些消息的可用性增加了重建队列后恢复联机消息的可能性。

例如，在此图中，Replication Server TOKYO_RS 具有到 MANILA_RS 的直接路由，MANILA_RS 具有到 SYDNEY_RS 的直接路由。

图 24：保存间隔示例



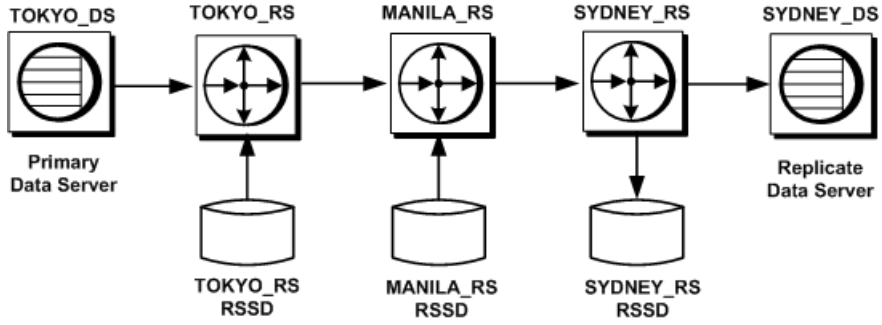
TOKYO_RS 会在 MANILA_RS 接收到消息后仍将这些消息保留一段时间。如果 MANILA_RS 遇到分区故障，它将会要求 TOKYO_RS 重新发送积压的消息。MANILA_RS 也可以保留消息，以便允许 SYDNEY_RS 从故障中恢复。

如果存储在某个稳定队列段上的所有消息的存储时间都至少达到了 *save_interval* 设置的时间，Replication Server 会删除该段，以便可以重新使用它。

为路由设置保存间隔

可以使用 *save_interval* 参数在源 Replication Server 上执行 **alter route** 命令，以便为路由设置保存间隔。

图 25：保存间隔示例



例如，要将 Replication Server TOKYO_RS 设置为将发往 MANILA_RS 的任何消息保存一个小时，请输入：

```
alter route to MANILA_RS
    set save_interval to '60'
```

缺省情况下，*save_interval* 设置为 0（分钟）。对于容量很小的系统，这可能是一个可以接受的恢复设置，因为 Replication Server 不会在接收到目标服务器的确认后立即删除消息，而是定期删除大块消息。

不过，要协调从 Replication Server 接收分配数据的节点的容量和活动，并增加从数据库或分区故障中完全恢复的可能性，您可能需要更改 *save_interval* 设置。

设置的时间一定要够用，以确保在稳定队列中发生分区故障时有足够的时间来恢复您的系统。还要考虑分配给积压消息的分区大小。分区必须足够大，以存放额外消息。

有关确定队列空间要求的帮助，请参见《Replication Server 设计指南》中的容量规划指南。

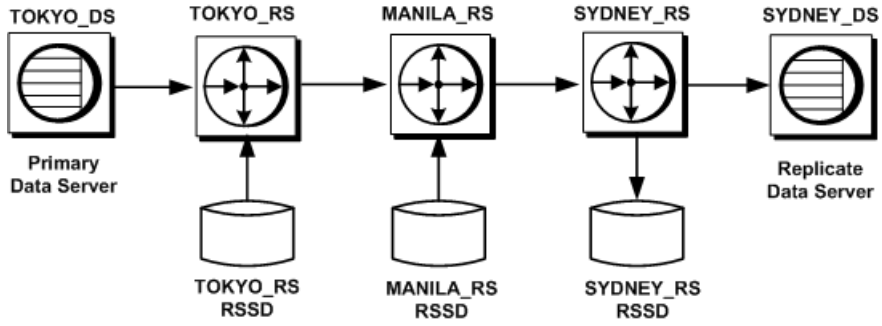
Replication Server 与数据服务器之间的连接

您可以为 Replication Server 之间的连接设置保存间隔以进行消息恢复。

当您为 Replication Server 与数据服务器和数据库之间的物理或逻辑连接设置了 *save_interval* 时，您就允许 Replication Server 将事务保存在 DSI 队列中。可以使用 **sysadmin restore_dsi_saved_segments** 恢复积压的事务。请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**sysadmin restore_dsi_saved_segments**”。

从事务转储和数据库转储将数据库装载至先前的状态后，您可以使用这些保存的事务来重新同步该数据库。

图 26：保存间隔示例



例如，在此图中，如果连接到 Replication Server SYDNEY_RS 的复制数据服务器 SYDNEY_DS 发生故障，它可以在恢复后获取保存在 SYDNEY_RS 的 DSI 队列中的消息，以重新同步复制数据库。

您也可以使用 *save_interval* 为存放着一些复制数据或接收应用函数的数据库设置热备份。

为连接设置保存间隔

可以使用 *save_interval* 参数在 Replication Server 上执行 **alter connection** 命令，以便为数据库连接设置保存间隔。

例如，要将 Replication Server SYDNEY_RS 设置为将发往复制数据服务器 SYDNEY_DS 的任何消息保存一小时，请输入：

```
alter connection to SYDNEY_DS.pubs2
    set save_interval to '60'
```

缺省情况下，*save_interval* 设置为 0（分钟）。

您还可以为逻辑连接的 DSI 队列和实现队列配置保存间隔。

另请参见

- 配置逻辑连接保存间隔 （第 93 页）

备份 RSSD

在执行任何复制 DDL（例如，更改路由或添加预订）之后执行 RSSD 的转储。

如果您无法恢复 RSSD 的最新状态，则恢复可能会很复杂。您使用的过程取决于自上次转储以来发生的 RSSD 活动的数量。

另请参见

- 从转储中恢复 RSSD 的过程 （第 299 页）

创建协调的转储

当您必须通过恢复备份来恢复主数据库时，还必须确保位于其它节点的受影响的数据库中的复制数据与主数据一致。

为在恢复后能在多个数据服务器上保持一致，Replication Server 提供了一种方法，用于在复制系统中的所有节点上协调数据库转储和事务转储。

您要从主数据库启动一个数据库转储或事务转储。RepAgent 会从日志检索转储记录，并将其提交到 Replication Server，以便可以将转储请求分配给复制节点。此方法可以确保将所有数据恢复到已知的一致状态。

您只能对存储主数据或复制数据的数据库使用协调的转储，而不能对既存储主数据又存储复制数据的数据库使用。您可以从主数据库内启动协调的转储。

协调转储的工作流程如下：

- 每个节点的“复制系统管理员”在指派给相关数据库的每个函数字符串类中为 **rs_dumpdb** 和 **rs_dumptran** 系统函数创建函数字符串。这些函数字符串应该调用执行 **dump database** 和 **dump transaction** 或等效命令的存储过程，并更新 `rs_lastcommit` 系统表。有关示例，请参见《Replication Server 参考手册》。
- 您必须使用可以在其中创建和修改函数字符串的函数字符串类，例如派生类。
- 每个复制节点的复制系统管理员使用 **alter connection** 命令将 Replication Server 配置为启用协调的转储。
- 在主数据库中启动转储后，RepAgent 会将 **dump database** 或 **dump transaction** 日志记录传输给 Replication Server。
- Replication Server 将 **rs_dumpdb** 或 **rs_dumptran** 函数调用分配给具有对数据库中已复制表的预订的节点。
- 复制节点上的 **rs_dumpdb** 和 **rs_dumptran** 函数字符串在每个复制节点执行自定义的存储过程。

另请参见

- 管理函数字符串类 （第 23 页）

从分区丢失或分区故障中恢复

当 Replication Server 检测到分区发生故障或分区丢失时，它会关闭正使用该分区的稳定队列，并会记录有关故障的消息。重新启动 Replication Server 不能改正此问题。您必须删除已损坏的分区并重建稳定队列。

是否能够完全恢复取决于从队列中清除的消息量以及故障发生后您应用恢复过程的快慢。如果 Replication Server 在复制系统中保持最短的延迟时间，则重建其队列时，只有最新的消息会丢失。

如果主 Replication Server 中发生分区故障，您通常可以使用脱机的数据库日志重新从消息源发送丢失的消息。如果复制 Replication Server 中发生分区故障，则需要从上游 Replication Server 的稳定队列中进行恢复。

在某些情况下，使用脱机日志可能是唯一能够恢复您的消息的方法。如果 Replication Server 已挂起路由或连接，或者，网络或数据服务器连接已断开，积压的消息可能已堆积在 Replication Server 的稳定队列中。除非您所指定的保存间隔设置能够解决积压问题，否则恢复这些消息的可能性会随着时间的推移而减小。源 Replication Server 可能已经将消息从其稳定队列中删除，并且可能已经截断了数据库日志。

注意： 您可以设置保存间隔以进行恢复。

另请参见

- 恢复的保存间隔 （第 286 页）

分区丢失或故障的症状和相关恢复过程

了解在何种情况下要使用相应的恢复过程来处理分区丢失或分区故障，以及在哪里可以找到这些过程。

表 31. 分区丢失或故障的症状和相关恢复过程

症状	使用此过程
Replication Server 可以检测到丢失、损坏或发生故障的稳定队列。	从分区丢失或分区故障中恢复。
由于发生故障的 Replication Server 中存在积压，而前一节点保存的消息不足，所以会发生消息丢失。	从脱机数据库日志恢复消息。
除丢失消息外，数据库日志也已被截断。辅助截断点无效，或者执行了 **dbcc settrunc('ltm', 'ignore')** 命令，以便将尚未由 RepAgent 传送到 Replication Server 的日志记录截断。	可以使用“从数据库日志恢复截断的消息”来恢复数据库日志。然后使用“从脱机数据库日志恢复消息”来重建稳定队列并恢复丢失的消息。

另请参见

- 从分区丢失或分区故障中恢复 （第 291 页）
- 从脱机数据库日志中恢复消息 （第 292 页）
- 从截断的主数据库日志中恢复消息 （第 295 页）

从分区丢失或分区故障中恢复

在 Replication Server 检测到丢失、损坏或发生故障的稳定队列时，从 Replication Server 分区丢失或分区故障中恢复。

1. 登录到 Replication Server，然后删除发生故障的分区：

```
drop partition logical_name
```

Replication Server 不会立即删除正在使用的分区。如果分区未损坏，只有在传递并删除了该分区中存放的所有消息之后，Replication Server 才会删除该分区。请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**drop partition**”。

2. 如果发生故障的分区是 Replication Server 唯一可用的分区，请再添加另一个分区来替换该分区：

```
create partition logical_name
on 'physical_name' with size size
[starting at vstart]
```

请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**create partition**”。

3. 因为该分区已损坏，所以您必须重建稳定队列：

```
rebuild queues
```

删除了分区中的所有稳定队列后，Replication Server 会从系统中删除发生故障的分区，然后使用其余的分区联机重建队列。

4. 重建队列后，请检查 Replication Server 日志中是否有丢失检测消息。
5. 如果 Replication Server 检测到消息丢失，请执行以下操作之一：
   - 执行从脱机数据库日志中恢复消息
   - 请求 Replication Server 忽略丢失，方法是：在检测到丢失的 Replication Server 上对数据库执行 **ignore loss** 命令。

下一

如果您指定 Replication Server 忽略消息丢失，而且您已经重建了 Replication Server 队列（该 Replication Server 是路由的一部分），请在目标重新创建预订或使用带有 **-r** 标志的 **rs_subcmp** 程序来调和主数据和复制数据。

另请参见

- 联机重建队列 （第 311 页）
- 重建稳定队列后进行丢失检测 （第 313 页）
- 从脱机数据库日志中恢复消息 （第 292 页）

从脱机数据库日志中恢复消息

发生分区故障后从脱机日志中恢复消息。

如果联机日志中没有恢复所需的所有数据，您必须将一个旧版本的主数据库装载到一个单独的数据库中，然后为该数据库启动 RepAgent。虽然 RepAgent 访问的是不同的数据库，但它提交的消息就如同来自于要恢复其消息的数据库。

1. 通过使用 **-M** 标志，在独立模式下重新启动 Replication Server。
2. 重建稳定队列。登录到 Replication Server，然后输入：

```
rebuild queues
```

3. 在每个节点中检查 Replication Server 日志中是否有“Checking Loss”消息，并使用错误日志消息中的日期和时间确定要装载的转储。
4. 为临时恢复数据库启用 RepAgent：

   ```
   sp_config_rep_agent temp_dbname, 'enable', \
   'rs_name', 'rs_user_name', 'rs_password'
   ```

   请参见《Replication Server 管理指南第一卷》的“管理 RepAgent 和支持 Adaptive Server”中的“设置 RepAgent”。
5. 将数据库转储和第一个事务日志转储装载到临时恢复数据库中。
6. 在恢复模式下为临时数据库启动 RepAgent：

   ```
   sp_start_rep_agent temp_dbname, 'recovery', \
   'connect_dataserver', 'connect_database', \
   'rs_name', 'rs_user_name', 'rs_password'
   ```

   其中，*'connect_dataserver'* 和 *'connect_database'* 指定原始主数据服务器和数据库。

   RepAgent 会将临时恢复数据库事务日志中的数据传送到原始主数据库。当 RepAgent 完成对事务日志的扫描后，它会关闭。
7. 验证 RepAgent 是否已重放了临时数据库的事务日志。请使用以下任一方法：

   - 检查 Adaptive Server 日志中是否有类似以下内容的消息：

     ```
     Recovery of transaction log is complete. Please
     load the next transaction log dump and then start
     up the Rep Agent Thread with sp_start_rep_agent,
     with ‘recovery’ specified.
     ```

     然后，执行适当的操作。
   - 从 Adaptive Server 中，执行：

     ```
     sp_help_rep_agent dbname, 'recovery'
     ```

     本过程显示 RepAgent 的恢复状态。如果恢复状态为“not running”或“end of log”，则说明恢复已完成。您可以装载下一个事务日志转储。如果恢复状态为“initial”或“scanning”，则说明日志尚未重放或重放未完成。
8. 如果自上次执行数据库转储后又执行了其它恢复过程，则可能需要在装载事务日志转储后更改数据库生成号。
9. 如果有更多的事务日志转储要装载，请对每个转储重复以下三个步骤：
   a) 装载下一个事务日志转储。（一定要按正确顺序装载转储。）
   b) 在恢复模式下重新启动 RepAgent。
   c) 查看 Adaptive Server 日志中是否有完成消息，或使用 **sp_help_rep_agent**。
10. 检查 Replication Server 日志中是否有丢失检测消息。

    除非您未能将数据库装载到足以检索所有消息的以前状态，否则应该检测不到丢失。
11. 在正常模式下重新启动 Replication Server。

12. 在正常模式下，为原始主数据服务器和数据库重新启动 RepAgent。

另请参见

- 联机重建队列 （第 311 页）
- 确定要装载的转储 （第 318 页）
- 确定数据库生成号 （第 319 页）
- 重建稳定队列后进行丢失检测 （第 313 页）

从联机数据库日志中恢复消息

恢复仍在主数据库的联机日志中的消息。

1. 停止所有客户端活动。
2. 在恢复模式下为主数据库重新启动 RepAgent。

   此过程会导致 RepAgent 从头扫描日志，所以它会检索所有消息。

从截断的主数据库日志中恢复

从在 Replication Server 收到消息之前截断主事务日志所导致的故障中恢复。

如果 RepAgent、Replication Server（管理主数据库）或它们之间的网络已断开很长时间，而且 RepAgent 或 Replication Server 无法从事务日志读取记录，通常就会发生这种情况。辅助截断点是无法移动的，这可以防止 Adaptive Server 截断日志，并使得主数据库的事务日志填满。稍后，您可以通过先后执行 **dbcc settrunc (ltm, ignore)** 和 **sp_stop_rep_agent** 来删除辅助截断点。

当发生故障的组件恢复服务后，Replication Server 上会丢失消息。根据丢失消息的状态不同，请使用以下过程之一：

- 如果消息仍在主数据库的联机日志中（这种情况不太可能发生），请从联机数据库日志中恢复消息。
- 如果已从联机数据库日志中截断消息，请从数据库日志中恢复截断的消息。
  在此过程中，您必须将先前的数据库转储和事务日志转储装载到一个临时恢复数据库。然后，将 RepAgent 与该数据库连接，以便向 Replication Server 传送截断的日志。在恢复了丢失的日志记录后，您可以使用常规主数据库来重新启动系统。使用临时恢复数据库允许从在主数据库日志被截断后仍继续使用主数据库的客户端恢复事务。

  注意： 临时数据库专门用于恢复消息。对该数据库的任何修改都将妨碍装载下一个事务日志转储。还应限制原始主数据库上的活动，以便在必须再次转储和截断原始主数据库上的事务日志之前能够完成恢复。

另请参见

- 从联机数据库日志中恢复消息 （第 294 页）

从截断的主数据库日志中恢复消息

重放脱机事务日志以从主数据库日志中恢复截断的消息。

1. 创建一个临时数据库，使原始数据库和临时数据库中的 **sysusages** 表类似。

   为此，您必须在创建临时数据库时使用与创建原始数据库时的所使用的相同的 **create database** 和 **alter database** 命令序列。
2. 关闭 Replication Server。
3. 通过使用 **-M** 标志，在独立模式下重新启动 Replication Server。
4. 登录到 Replication Server，然后对要恢复的每个主数据库执行 **set log recovery** 命令。

   此命令会将 Replication Server 置于数据库的丢失检测模式。Replication Server 会记录类似以下内容的消息：

   ```
   Checking Loss for DS1.PDB from DS1.PDB
       date=Nov-01-1995 10:35am
       qid=0x01234567890123456789
   ```
5. 执行 **allow connections** 命令，以便允许 Replication Server 只在恢复模式下接受来自其它 Replication Server 和来自 RepAgent 的连接。

   注意： 如果通过使用脚本在正常模式下自动重新启动 RepAgent 来尝试连接到此 Replication Server，Replication Server 会拒绝此连接。您必须在恢复模式下重新启动 RepAgent，同时指向正确的脱机日志。这一步骤允许您在当前事务得到处理之前重新发送旧的事务日志。
6. 将数据库转储装载到临时主数据库中。
7. 将第一个或下一个事务日志转储装载到临时主数据库。
8. 在恢复模式下为临时数据库启动 RepAgent：

   ```
   sp_start_rep_agent temp_dbname, 'recovery',
   'connect_dataserver', 'connect_database',
   'repserver_name', 'repserver_username',
   'repserver_password'
   ```

   其中，*connect_dataserver* 和 *connect_database* 指定原始主数据服务器和数据库。

   RepAgent 会将临时恢复数据库事务日志中的数据传送到原始主数据库。当 RepAgent 完成对当前事务日志的扫描后，它会关闭。
9. 通过执行以下操作之一，验证 RepAgent 是否已重放了临时数据库的事务日志：

   - 检查 Adaptive Server 日志中是否有以下消息：

     ```
     Recovery of transaction log is complete. Please
     load the next transaction log dump and then start
     up the Rep Agent Thread with sp_start_rep_agent,
     with ‘recovery’ specified.
     ```

     然后执行相应的操作。

- 执行 **admin who_is_down**。
  如果 RepAgent 报告“down”，请装载下一个事务日志。

10. 重复第 7 步到 9 步，直到处理完所有事务日志。

    现在就可以从主数据库恢复正常复制了。

11. 关闭仍处于独立模式的 Replication Server。
12. 您可能需要执行 **rs_zeroltm** 以清除定位符信息。

```
rs_zeroltm data_server, database
dbcc settrunc('ltm', 'valid')
```

13. 在正常模式下重新启动 Replication Server。
14. 使用 **sp_start_rep_agent** 重新启动主数据库和 RSSD 的 RepAgent。
15. 如果自上次执行数据库转储后又执行了其它恢复过程，则可能需要在装载事务日志转储后更改数据库生成号。

另请参见

- 为数据库设置日志恢复 （第 316 页）
- 确定数据库生成号 （第 319 页）

从主数据库故障中恢复

如果主数据库发生故障，您无法恢复所有提交的事务，则必须将该数据库装载到先前的状态，并执行为在复制节点进行一致性恢复而设计的恢复过程。

大多数数据库故障可以在不丢失任何提交事务的情况下恢复。如果可以通过重新启动来恢复数据库，则不需要任何特别的 Replication Server 恢复过程。Replication Server 将会与数据库进行“握手”，以确保复制系统中没有丢失任何事务，也没有任何重复的事务。

以下是从主数据库故障中恢复的两种可能情况：

- 仅使用主转储进行恢复。
  如果您没有协调的转储，则可以装载发生故障的主数据库，然后验证复制数据库与恢复的主数据库是否一致。
- 使用协调的转储进行恢复。
  如果您有主数据库和复制数据库的协调的转储，您就可以使用它们将复制系统中的所有数据库装载为一致的状态。

从转储装载主数据库

如果仅在复制系统中装载主数据库，请将该数据库装载到以前的状态，并解决与复制数据库之间的任何不一致问题。

1. 登录到主 Replication Server 以获取主数据库的数据库生成号：

```
admin get_generation, data_server, database
```

记下该生成号，以后的步骤需要使用该生成号。

2. 关闭主数据库的 RepAgent：

```
sp_stop_rep_agent database
```

3. 挂起到主数据库的 DSI 连接（专用）。
4. 将数据库装载到最近或以前的状态。

   此步骤要求装载最近的数据库转储以及所有后续的事务日志转储。

   有关说明，请参见《Adaptive Server Enterprise 系统管理指南》。
5. 恢复 DSI 连接。
6. 转储事务日志。

   输入：

```
use database
go
dbcc settrunc('ltm', 'ignore')
go
dump tran database with truncate_only
go
dbcc settrunc('ltm', 'valid')
go
```

7. 在恢复的主数据库中执行 **dbcc settrunc** 命令，以将生成号设置为下一个较大的号码。

   例如，如果步骤 1 中的 **admin get_generation** 命令返回 0，请输入：

```
use database
go
dbcc settrunc('ltm', 'gen_id', 1)
```

8. 清除定位符信息。

   输入：

```
rs_zeroltm data_server, database
```

9. 启动主数据库的 RepAgent。为此，请执行以下命令：

```
sp_start_rep_agent database
```

10. 在复制节点为每个预订运行 **rs_subcmp** 程序。使用 **-r** 标志来调和复制数据和主数据，或删除所有预订并重新创建它们。

    请参见《Replication Server 管理指南第一卷》的“管理预订”的“获取预订信息”的“验证预订一致性”的“使用 **rs_subcmp** 来查找和更正不一致”以及《Replication Server 参考手册》的“可执行程序”中的“**rs_subcmp**”。

从协调的转储装载

只有在您同时拥有主数据库和复制数据库的协调的转储时，才能使用此过程。此过程将主数据库和所有复制数据库装载为同一状态。

1. 执行从转储装载主数据库的过程中的第 1 步到第 10 步。

2. 挂起必须恢复的复制数据库的连接。
3. 对于每个复制数据库，登录到管理它的 Replication Server，然后挂起到该数据库的连接。

   输入：

   ```
   suspend connection to data_server.database
   ```
4. 从协调的转储装载与恢复的主数据库状态对应的复制数据库。
5. 对于每个复制数据库，登录到管理它的 Replication Server，然后执行 **sysadmin set_dsi_generation** 命令，以便将该数据库的生成号设置为与步骤 1 中使用的生成号相同。

   输入：

   ```
   sysadmin set_dsi_generation, 101,
       primary_data_server, primary_database,
       replicate_data_server, replicate_database
   ```

   参数 *primary_data_server* 和 *primary_database* 指定要装载的主数据库。参数 *replicate_data_server* 和 *replicate_database* 指定要装载的复制数据库。

   以这种方式设置生成号可以防止 Replication Server 将队列中可能存在的任何旧消息应用于复制数据库。
6. 对于每个复制数据库，登录到管理它的 Replication Server，然后恢复到该数据库的连接以重新启动该数据库的 DSI。

   输入：

   ```
   resume connection to data_server.database
   ```
7. 在正常模式下重新启动主 Replication Server。
8. 在正常模式下为主数据库重新启动 RepAgent。

下一

对于发生故障时正在实现的任何预订，请将其删除，然后重新创建它们。

另请参见

- 从转储装载主数据库 （第 296 页）

从 RSSD 故障中恢复

如果您无法恢复 RSSD 的最新数据库状态，则从 RSSD 故障中恢复将是一个很复杂的过程。在这种情况下，您必须从旧的数据库转储和事务日志转储装载 RSSD。

注意： 无法通过使用跨平台 **dump** 和 **load** 或使用 **bcp** 等来实现 RSSD 数据库的跨平台迁移。若要迁移，必须在新平台上重新构建复制系统。

恢复 RSSD 的过程与恢复主数据库的过程类似。但是，该过程需要更多的步骤，因为 RSSD 存放着有关复制系统本身的信息。RSSD 系统表与复制系统中稳定队列和其它 RSSD 的状态紧密相关。

如果某个 Replication Server 的 RSSD 发生故障，您首先需要确定要进行恢复的范围。为此，请执行以下一个或多个操作：

- 在 RSSD 可用后，登录到 Replication Server，然后执行 **admin who_is_down**。在 RSSD 处于不活动状态期间，某些 Replication Server 线程可能已关闭。
  - 如果入站（或出站）队列或 RSI 出站队列的 SQM 线程已关闭，请重新启动 Replication Server。
  - 如果 DSI 线程已关闭，请重新开始到相关数据库的连接。
  - 如果 RSI 线程已关闭，请重新开始到目标数据库的路由。
- 使用 **sp_help_rep_agent** 系统过程检查所有连接的 RepAgent，查看它们是否正在运行。（由于 RSSD 关闭而导致的错误可能已使 RepAgent 关闭。）如有必要，重新启动它们。
- 如果您无法恢复 RSSD 的最新数据库状态，则必须从旧的数据库转储和事务日志转储来装载 RSSD。

另请参见

- 从转储中恢复 RSSD 的过程 （第 299 页）

从转储中恢复 RSSD 的过程

用于恢复 RSSD 的过程取决于自上次 RSSD 转储以来 RSSD 活动的数量。RSSD 故障有四个从低到高的严重级别，每个级别都具有相应的恢复要求。

表 32. 从 RSSD 故障中恢复的过程

自上次 RSSD 转储以来的活动	使用此过程
无 DDL 活动	基本 RSSD 恢复过程。
有 DDL 活动，但未创建新的路由或预订	预订比较过程
有 DDL 活动，但未创建新的路由	预订比较过程
创建了新路由	拆散/重聚过程

另请参见

- 使用基本 RSSD 恢复过程 （第 300 页）
- 使用预订比较过程 （第 302 页）
- 使用预订重新创建过程 （第 308 页）
- 使用拆散和重聚过程 （第 310 页）

使用基本 RSSD 恢复过程

自上次 RSSD 转储以来未执行任何 DDL 命令时恢复 RSSD。RCL 中的 DDL 命令包括那些用于创建、更改或删除路由、复制定义、预订、函数字符串、函数、函数字符串类或错误类的命令。

其它 RSSD 恢复过程也参照此过程中的某些步骤。

警告！ 在您完成此恢复过程以前，请不要执行任何 DDL 命令。

1. 关闭所有连接到当前 Replication Server 的 RepAgent。
2. 因为其 RSSD 已发生故障，所以当前 Replication Server 已关闭。如果由于某种原因当前 Replication Server 未关闭，请登录到该 Replication Server，然后使用 **shutdown** 命令将其关闭。

   注意： 某些消息可能仍存在于 Replication Server 稳定队列中。当您在后面的步骤中重建这些队列时，这些队列中的数据可能会丢失。
3. 通过装载最近的 RSSD 数据库转储和所有事务转储来恢复 RSSD。
4. 使用 **-M** 标志，在独立模式下重新启动 Replication Server。

   您必须在独立模式下启动 Replication Server，因为此时稳定队列与 RSSD 状态不一致。当 Replication Server 在独立模式下启动时，不会自动激活对稳定队列的读取。
5. 登录到 Replication Server，然后获取 RSSD 的生成号。

   输入：

   ```
   admin get_generation, data_server, rssd_name
   ```

   例如，Replication Server 可能会返回生成号 100。
6. 在 Replication Server 中，重建队列。

   输入：

   ```
   rebuild queues
   ```
7. 在恢复模式下启动所有连接到当前 Replication Server 的 RepAgent（RSSD RepAgent 除外）。

   输入：

   ```
   sp_start_rep_agent dbname, recovery
   ```

   等到每个 RepAgent 都在 Adaptive Server 日志中记录一条消息，以说明它已完成当前的日志记录。
8. 检查该 Replication Server 日志以及所有具有源自当前 Replication Server 的直接路由的 Replication Server 日志中的丢失消息。
   - 如果发生故障时您的所有路由都处于活动状态，您可能不会遇到任何实际的数据丢失。

- 但是，丢失检测可能会指示有实际丢失。如果数据库日志在主数据库被截断，导致重建过程没有足够的信息进行恢复，则可能会检测到实际数据丢失。如果您遇到实际数据丢失，请使用从截断的主数据库日志中进行恢复的过程从旧的转储中重装数据库日志。

9. 关闭由当前 Replication Server 管理的所有主数据库的 RepAgent。

输入：

```
sp_stop_rep_agent dbname
```

10. 关闭 Replication Server。

11. 向上移动到辅助截断点。

在 Adaptive Server 上为恢复的 RSSD 执行 **dbcc settrunc** 命令：

```
use rssd_name
go
dbcc settrunc('ltm', 'ignore')
go
dump tran rssd_name with truncate_only
go
begin tran commit tran
go 40
```

注意： **begin tran commit tran go 40** 命令会将 Adaptive Server 日志移动到下一页。

12. 清除定位符信息。

输入：

```
rs_zeroltm rssd_server, rssd_name
go
```

13. 在 Adaptive Server 上为恢复的 RSSD 执行 **dbcc settrunc** 命令，以将生成号设置为比步骤 5 中的 **admin get_generation** 返回的号码大一的生成号。

输入：

```
dbcc settrunc ('ltm', 'gen_id', generation_number)
go
dbcc settrunc('ltm', 'valid')
go
```

如有必要，请对此生成号和当前时间进行记录，以便您可以返回到这个 RSSD 恢复过程。或者，您可以在设置该生成号之后转储数据库。

14. 在正常模式下重新启动 Replication Server。

如果您作为预订比较或预订重新创建过程的一部分执行了此过程，上游 RSI 出站队列可能会包含已使用 **rs_subcmp** 应用了的事务（这些事务发往当前 Replication Server 的 RSSD）。如果是这种情况，启动了 Replication Server 之后，错误日志可能会包含有关重复插入的警告。您可以安全地忽略这些警告。

15. 在正常模式下为 RSSD 和用户数据库重新启动 RepAgent。

如果您将此过程作为预订比较或预订重新创建 RSSD 恢复过程的一部分执行，应该能够看到有关 RSSD 丢失的消息，这些消息是所有具有来自当前 Replication Server 的路由的 Replication Server 中检测到的。

另请参见

- 联机重建队列 （第 311 页）
- 从截断的主数据库日志中恢复 （第 294 页）
- 重建稳定队列后进行丢失检测 （第 313 页）

使用预订比较过程

如果自上次事务转储以来执行了某些 DDL 命令，但未创建任何新的预订或路由，请按照此 RSSD 恢复过程进行操作。

RCL 中的 DDL 命令包括那些用于创建、更改或删除路由、复制定义、预订、函数字符串、函数、函数字符串类或错误类的命令。

警告！ 在您完成此恢复过程以前，请不要执行任何 DDL 命令。

按照此过程进行操作可使发生故障的 RSSD 与上游 RSSD 一致，或与最近的数据库和事务转储一致（如果没有上游 Replication Server）。继而会使下游 RSSD 与恢复的 RSSD 一致。

如果自上次事务转储以来在当前 Replication Server 上执行了 DDL 命令，您可能必须重新执行这些命令。

警告！ 如果您的操作环境是混合版本环境（即，您的复制系统中的 Replication Server 的版本级别不完全相同），则该过程可能会失败。

1. 若要为发生故障的 RSSD 进行恢复准备，请执行基本 RSSD 恢复过程的第 1 步到第 4 步。
2. 要为所有上游 RSSD 进行恢复准备，请在每个上游 Replication Server 上执行 **admin quiesce_force_rsi** 命令。
   - 此步骤可确保在您执行 **rs_subcmp** 程序之前，所有来自当前 Replication Server 的已提交事务均已应用。
   - 从当前 Replication Server 的最上游 Replication Server 开始，按顺序执行此命令。
   - 确保已应用了 RSSD 更改，即，RSSD DSI 出站队列为空。
   - 紧邻当前 Replication Server 的上游 Replication Server 不能停顿。
3. 要为所有下游 RSSD 进行恢复准备，请在每个下游 Replication Server 上执行 **admin quiesce_force_rsi** 命令。
   - 此步骤可确保在您执行 **rs_subcmp** 程序之前，所有发往当前 Replication Server 的已提交事务均已应用。

- 从紧邻当前 Replication Server 的下游 Replication Server 开始，按顺序执行此命令。
- 确保已应用了 RSSD 更改，即，RSSD DSI 出站队列为空。

4. 使用 **rs_subcmp** 程序，调和发生故障的 RSSD 和所有上游 RSSD。
   - 先在没有进行调和的情况下执行 **rs_subcmp**，以了解它将要执行的操作。在您准备好进行调和后，请使用 **-r** 标志来调和复制数据和主数据。
   - 您必须以维护用户的身份执行 **rs_subcmp**。有关维护用户的详细信息，请参见《Replication Server 管理指南第一卷》中的“管理 Replication Server 安全性”。
   - 在每个实例中，将发生故障的 RSSD 指定为复制数据库。
   - 在每个实例中，将每个上游 Replication Server 的 RSSD 指定为主数据库。
   - 从最上游的 Replication Server 开始，一直向下到所有具有到当前 Replication Server 的路由（直接或间接）的其它 Replication Server。
   - 调和以下各 RSSD 系统表：`rs_articles`、`rs_classes`、`rs_columns`、`rs_databases`、`rs_erroractions`、`rs_functions`、`rs_funcstrings`、`rs_objects`、`rs_publications`、`rs_systext` 和 `rs_whereclauses`。
   - 当您对复制 RSSD 表执行 **rs_subcmp** 时，**select** 语句的 **where** 和 **order by** 子句必须包括所有要复制的行。

     现在，发生故障的 RSSD 应该已恢复。
5. 使用 **rs_subcmp** 程序，调和所有下游 RSSD 和当前 Replication Server 的 RSSD（该 RSSD 已在上一步中恢复）。
   - 先在没有进行调和的情况下执行 **rs_subcmp**，以了解它将要执行的操作。在您准备好进行调和后，请使用 **-r** 标志来调和复制数据和主数据。
   - 您必须以维护用户的身份执行 **rs_subcmp**。有关维护用户的详细信息，请参见《Replication Server 管理指南第一卷》中的“管理 Replication Server 安全性”。
   - 在每个实例中，将恢复的 RSSD 指定为主数据库。
   - 在每个实例中，将每个下游 Replication Server 的 RSSD 指定为复制数据库。
   - 从紧邻的下游 Replication Server 开始，一直向下到所有具有源自当前 Replication Server 的路由（直接或间接）的其它 Replication Server。
   - 调和以下各 RSSD 系统表：`rs_articles`、`rs_classes`、`rs_columns`、`rs_databases`、`rs_erroractions`、`rs_functions`、`rs_funcstrings`、`rs_objects`、`rs_publications`、`rs_systext` 和 `rs_whereclauses`。
   - 当您对复制 RSSD 表执行 **rs_subcmp** 时，**select** 语句的 **where** 和 **order by** 子句必须选择所有要复制的行。

     现在，所有下游 RSSD 应该已完全恢复。
6. 如果恢复的 Replication Server 是 ID Server，您必须在其 RSSD 中恢复 Replication Server 和数据库 ID。
   a) 对于每个 Replication Server，在 `rs_databases` 和 `rs_sites` 系统表中检查它们的 ID。

b) 如果恢复的 RSSD 的 rs_idnames 系统表中的某些行丢失，请插入相应的行。
c) 从要恢复的 RSSD 的 rs_idnames 系统表中删除任何已不再属于复制系统的数据库或 Replication Server 的 ID。
d) 确保 rs_ids 系统表是一致的。

在当前 Replication Server 的 RSSD 中执行以下存储过程：

```
rs_mk_rsids_consistent
```

7. 如果要恢复的 Replication Server 不是 ID Server，而且 在上次事务转储后，在要恢复的 Replication Server 上创建了数据库连接，请在 ID Server 的 RSSD 中，从 rs_idnames 系统表删除与该数据库连接对应的行。
8. 执行基本 RSSD 恢复过程的第 5 步到第 14 步。
9. 要完成 RSSD 恢复，请重新执行自上次事务转储以来在当前 Replication Server 上执行的任何 DDL 命令。

另请参见

- 用于复制的 RSSD 系统表上的 rs_subcmp 的 select 语句 （第 304 页）
- 使用基本 RSSD 恢复过程 （第 300 页）

用于复制的 RSSD 系统表上的 rs_subcmp 的 select 语句

在RSSD恢复过程中对复制的RSSD表执行 **rs_subcmp** 时，请明确地表示 **select** 语句的 **where** 和 **order by** 子句，以选择必须为每个系统表复制的所有行。

在列出的 **select** 语句中，*sub_select* 表示下面的子选择语句，该子选择语句选择作为当前 Replication Server 的源 Replication Server 的所有节点 ID：

```
(select source_rsid from rs_routes
    where
      (through_rsid = PRS_site_ID
      or through_rsid = RRS_site_ID)
    and
    dest_rsid = RRS_site_ID)
```

其中，*PRS_site_ID* 是为 **rs_subcmp** 操作管理主 RSSD 的 Replication Server 的节点 ID，*RRS_site_ID* 是为该操作管理复制 RSSD 的 Replication Server 的节点 ID。

对于 rs_columns、rs_databases、rs_funcstrings、rs_functions 和 rs_objects 系统表，如果 rowtype = 1，则该行为复制的行。只有复制的行需要使用 **rs_subcmp** 进行比较。

对于每个系统表，*primary_keys* 是其唯一索引。有关这些表的详细信息，请参见《Replication Server 参考手册》中的“Replication Server 系统表”。

注意： 以下是 **select** 语句的一般格式。在混合版本环境中，您可能需要调整这些 **select** 语句。

表 33. rs_subcmp 过程的 select 语句的一般格式

RSSD 表名	select 语句
rs_articles	select * from rs_articles,rs_objects where rs_objects.prsid in *sub_select* and rs_articles.objid = rsobjects.objid order by articleid
rs_classes	select * from rs_classes where prsid in *sub_select* order by classid
rs_columns	select * from rs_columns where prsid in *sub_select* and rowtype = 1 order by objid, basecolnum, colname, colnum, version
rs_databases	select * from rs_databases where prsid in *sub_select* and rowtype = 1 order by dbid, dbname, dsname, ldbid, ltype, ptype
rs_erroractions	select * from rs_erroractions where prsid in *sub_select* order by ds_errorid, errorclassid
rs_funcstrings	select * from rs_funcstrings where prsid in *sub_select* and rowtype = 1 order by fstringid
rs_functions	select * from rs_functions where prsid in *sub_select* and rowtype = 1 order by funcid, funcname, objid
rs_objects	select * from rs_objects where prsid in *sub_select* and rowtype = 1 order by active_inbound, dbid, has_baserepdef, objid, objname, objtype, phys_tablename, phys_objowner, version
rs_publications	select * from rs_publications where prsid in *sub_select* order by pubid

RSSD 表名	select 语句
rs_systext	select * from rs_systext where prsid in *sub_select* and texttype in ('O', 'S') order by parentid, texttype, sequence
rs_whereclauses	select * from rs_whereclauses,rs_articles,rs_objects where rs_objects.prsid in *sub_select* and rs_articles.objid = rsobjects.objid and rs_whereclauses.articleid = rs_artilces.articleid order by wclauseid

类和系统表

了解预订比较过程对类和系统表的影响，以及如何将 RSSD 保持一致的状态。

系统提供的函数字符串类和错误类最初没有指定的主节点，即，它们的节点 ID 等于 0。rs_default_function_class 类和 rs_db2_function_class 类是不能修改的，因而决不能有指定的主节点。可以为 rs_sqlserver_function_class 类和 rs_sqlserver_error_class 类指派主节点，并可对其进行修改。派生函数字符串类的主节点与其父类的主节点相同。

自上次事务转储后，如果要恢复的 Replication Server 作为函数字符串类或错误类的主节点，**rs_subcmp** 过程将在下游 RSSD 中找到孤行。

在这种情况下，请再次对 rs_classes、rs_erroractions、rs_funcstrings 和 rs_systext 系统表运行 **rs_subcmp**。设置 **prsid = 0** 以便为这些表重新填充必要的缺省设置。

例如，对 rs_classes 表使用以下 **select** 语句：

```
select * from rs_classes where prsid = 0
    order by primary_keys
```

示例

将 RSSD 保持一致的状态。

假定在您的复制系统中具有以下 Replication Server 节点，其中箭头 (→) 表示路由。节点 B 是出现故障的节点，而且没有间接路由。

- A > B
- C > B
- C > D
- B > E

这些 Replication Server 具有以下节点 ID：

- A = 1
- B = 2
- C = 3
- D = 4
- E = 5

在此示例中，要使 RSSD 达到一致状态，您要按照所示顺序对 rs_classes、rs_columns、rs_databases、rs_erroractions、rs_funcstrings、rs_functions、rs_objects 和 rs_systext 系统表执行以下任务。

调和上游 RSSD
调和系统表和上游 RSSD。

1. 对这些表运行 **rs_subcmp**，将节点 B 指定为复制节点，将节点 A 指定为主节点，并在 **where** 子句中指定 prsid = 1。

   例如，rs_columns 的 **select** 语句看起来应该类似于：

   ```
   select * from rs_columns where prsid in
       (select source_rsid from rs_routes
       where
           (through_rsid = 1 or through_rsid = 2)
            and dest_rsid = 2)
           and rowtype = 1
          order by objid, colname
   ```

2. 对以上各表运行 **rs_subcmp**，将节点 B 指定为复制节点，将节点 C 指定为主节点，并在 **where** 子句中指定 prsid = 3。

   例如，rs_columns 的 **select** 语句看起来应该类似于以下语句：

   ```
   select * from rs_columns where prsid in
         (select source_rsid from rs_routes
          where
            (through_rsid = 3 or through_rsid = 2)
             and dest_rsid = 2)
          and rowtype = 1
         order by objid, colname
   ```

调和下游 RSSD
调和系统表和下游 RSSD。
对以上各表运行 **rs_subcmp**，将节点 B 指定为主节点，将节点 E 指定为复制节点，并在 **where** 子句中指定 prsid = 2。

例如，rs_columns 的 **select** 语句看起来应该类似于：

```
select * from rs_columns where prsid in
      (select source_rsid from rs_routes
       where
         (through_rsid = 2 or through_rsid = 5)
          and dest_rsid = 5)
       and rowtype = 1
      order by objid, colname
```

请参见《Replication Server 参考手册》的“可执行程序”中的“**rs_subcmp**”和《Replication Server 参考手册》中的“Replication Server 系统表”。

使用预订重新创建过程

如果在上次事务转储后创建了新的预订或其它 DDL，并且未创建新的路由，请恢复需要重新创建丢失的预订的 RSSD。

RCL 中的 DDL 命令包括那些用于创建、更改或删除路由、复制定义、预订、函数字符串、函数、函数字符串类或错误类的命令。

警告! 在您完成预订重新创建恢复过程以前，请不要执行任何 DDL 命令。

此任务使发生故障的 RSSD 与上游 RSSD 保持一致，或与最近的数据库和事务转储保持一致（如果没有上游 Replication Server）。继而会使下游 RSSD 与恢复的 RSSD 一致。您还要删除或重新创建由于 RSSD 丢失而处于中间过渡状态的预订。

不过，在此过程中，

如果自上次事务转储以来在当前 Replication Server 上执行了 DDL 命令，您可能必须重新执行这些命令。

请参见《Replication Server 参考手册》的“可执行程序”中的“**rs_subcmp**”和《Replication Server 参考手册》中的“Replication Server 系统表”。

1. 若要为发生故障的 RSSD 进行恢复准备，请执行基本 RSSD 恢复过程的第 1 步到第 4 步。
2. 若要为所有上游和下游 Replication Server 进行恢复准备，请执行预订比较过程的第 2 步和第 3 步。
3. 关闭所有受上一步影响的上游和下游 Replication Server。使用 **shutdown** 命令。
4. 使用**-M** 标志，在独立模式下重新启动所有上游和下游 Replication Server。

   当您在独立模式下重新启动 Replication Server 时，所有连接到这些 Replication Server 的 RepAgent 会自动关闭。
5. 若要调和发生故障的 RSSD 和所有上游 RSSD，请执行预订比较过程的第 4 步。

   现在，发生故障的 RSSD 应该已恢复。
6. 若要调和所有下游 RSSD 和当前 Replication Server 的 RSSD，请执行预订比较过程的第 5 步。
7. 如果要恢复的 Replication Server 是 ID Server，要恢复其 RSSD 中的 ID，请执行预订比较过程的第 6 步。
8. 如果要恢复的 Replication Server 不是 ID Server，而且在上次事务转储后在要恢复的 Replication Server 上创建了数据库连接，请执行预订比较过程的第 7 步。
9. 在当前 Replication Server 的 `rs_subscriptions` 系统表中查询预订和复制定义或发布及其相关数据库的名称。

- 此外，还要查询所有预订当前 Replication Server 所管理的主数据的 Replication Server，或具有当前 Replication Server 所预订的主数据的 Replication Server。
- 您可以通过使用 **rs_helpsub** 存储过程来查询 `rs_subscriptions` 系统表。

10. 对于 `rs_subscriptions` 系统表中的每个用户预订，使用在第 9 步中获得的信息执行 **check subscription** 命令。

- 在当前 Replication Server 和所有预订了当前 Replication Server 管理的主数据的 Replication Server 上，或所有具有当前 Replication Server 预订的主数据的 Replication Server 上，执行此命令。
- 必须删除或重新创建不处于“有效”状态的预订，如下所述。

11. 对于每个具有非 VALID 预订的 Replication Server（当前 Replication Server 作为主服务器）：

- 注意其 `subid`，并从 `rs_subscriptions` 主系统表中删除相应的行。
- 使用 `rs_subscriptions` 中的 `subid` 在 `rs_rules` 系统表中找到相应的行，并且还要删除这些行。

对于 `rs_subscriptions` 和 `rs_rules` 系统表：

- 如果预订位于主表而且不在复制表中（因为它已被删除），则从主表中删除预订行。
- 如果预订位于复制表而且不在主表中，则从复制表中删除预订行。完成此过程的其余部分后，按照步骤 17 到 19 的说明，重新创建预订。
- 如果预订同时位于主表和复制表中，但在其中的一个节点处于非“有效”状态，则从两个表中删除这些行。完成此过程的其余部分后，按照步骤 17 到 19 的说明，重新创建预订。

12. 对于当前 Replication Server 具有非“有效”用户预订的每个主 Replication Server：

- 注意其 `subid`，并从 `rs_subscriptions` 主系统表中删除相应的行。
- 使用 `rs_subscriptions` 中的 `subid` 在 `rs_rules` 系统表中找到相应的行，并且还要删除这些行。

对于 `rs_subscriptions` 和 `rs_rules` 系统表：

- 如果预订位于主表而且不在复制表中，则从主表中删除预订行。完成此过程的其余部分后，按照步骤 17 到 19 的说明，重新创建预订。
- 如果预订位于复制表而且不在主表中（因为它已被删除），则从复制表中删除预订行。
- 如果预订同时位于主表和复制表中，但在其中的一个节点处于非“有效”状态，则从两个表中删除这些行。完成此过程的其余部分后，按照步骤 17 到 19 的说明，重新创建预订。

13. 对于已在步骤 17 到步骤 19 中删除的预订，在主 Replication Server 和复制 Replication Server 上，对其所有现有的实现队列执行 **sysadmin drop_queue** 命令。

14. 在正常模式下重新启动所有预订当前 Replication Server 所管理的主数据或具有当前 Replication Server 所预订的主数据的 Replication Server，及其 RepAgent。

15. 执行基本 RSSD 恢复过程的第 5 步到第 13 步。

16. 重新执行自上次事务转储以来在当前 Replication Server 上执行的任何 DDL 命令。
17. 为每个复制定义启用自动更正。
18. 使用批量实现方法或不实现重新创建丢失的预订。

    使用 **define subscription**、**activate subscription**、**validate subscription** 和 **check subscription** 命令进行批量实现。
19. 对于每个重新创建的预订，通过以下两种方法中的任一方法恢复主数据和复制数据之间的一致性：
    - 使用 **drop subscription** 命令和 **with purge** 选项删除预订。然后，重新创建预订。
    - 使用带有 **-r** 标志的 **rs_subcmp** 程序来调和复制预订数据和主预订数据。

另请参见

- 使用基本 RSSD 恢复过程 （第 300 页）
- 使用预订比较过程 （第 302 页）

使用拆散和重聚过程

如果自上次转储 RSSD 后创建了路由，您需要执行拆散和重聚过程。

1. 从复制系统删除当前 Replication Server。

   请参见《Replication Server 管理指南第一卷》中的“删除 Replication Server”。
2. 重新安装 Replication Server。

   有关重新安装 Replication Server 的完整信息，请参见适用于您的平台的 Replication Server 安装和配置指南。
3. 重新创建 Replication Server 路由和预订。

   请参见《Replication Server 管理指南第一卷》中的“管理路由”以及《Replication Server 管理指南第一卷》中的“管理预订”。

恢复支持任务

通过执行标准恢复任务，您可以在复制系统中操纵和标识关键数据。

恢复任务仅适用于与之对应的过程。

恢复支持任务包括：

- 重建稳定队列
- 重建稳定队列后，检查是否有 Replication Server 丢失检测消息
- 将 Replication Server 置于日志恢复模式并为数据库设置日志恢复
- 为数据库设置了日志恢复后，检查是否有 Replication Server 丢失检测消息

- 确定要装载的转储和日志
- 调整数据库生成号

另请参见

- 重建稳定队列 （第 311 页）
- 重建稳定队列后进行丢失检测 （第 313 页）
- 为数据库设置日志恢复 （第 316 页）
- 设置日志恢复后进行丢失检测 （第 317 页）
- 确定要装载的转储 （第 318 页）
- 调整数据库生成号 （第 318 页）

重建稳定队列

rebuild queues 命令将会删除所有现有队列，然后重建这些队列。该命令无法重建单个稳定队列。

根据您的具体情况，您可以联机或脱机重建队列。通常，您要首先联机重建队列，检测是否丢失了稳定队列消息。如果有消息丢失，您可以检索这些消息，方法是先将 Replication Server 置于独立模式，然后从脱机日志中恢复数据。

请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**rebuild queues**”。

联机重建队列

在联机重建过程中，Replication Server 处于正常模式。所有 RepAgent 和其它 Replication Server 都会自动断开与该 Replication Server 的连接。

连接尝试会被拒绝，并显示以下消息：

```
Replication Server is Rebuilding
```

Replication Server 和 RepAgent 会定期重试连接，直到 **rebuild queues** 完成。该命令完成后，连接将会成功。

清除队列后，重建就会完成。然后，Replication Server 会尝试从以下源中检索清除的消息：

- 具有到重建的 Replication Server 的直接路由的其它 Replication Server。如果您已在其它 Replication Server 上设置了保存间隔，则恢复的可能性会较大。
- Replication Server 管理的主数据库的数据库事务日志。

如果有丢失检索消息，您需要检查这些消息的状态。根据故障情况，如果已无法再从消息源获取这些消息，您可能需要使用脱机日志重建队列。或者，您可以请求 Replication Server 忽略丢失的消息。

另请参见

- 从脱机数据库日志中重建队列 （第 312 页）

- 重建稳定队列后进行丢失检测 （第 313 页）

从脱机数据库日志中重建队列

您可以从脱机数据库日志中恢复数据。

您只有在独立模式下重新启动了 Replication Server 后，才可以使用 **rebuild queues** 命令。在独立模式下执行 **rebuild queues** 会将 Replication Server 置于恢复模式中。

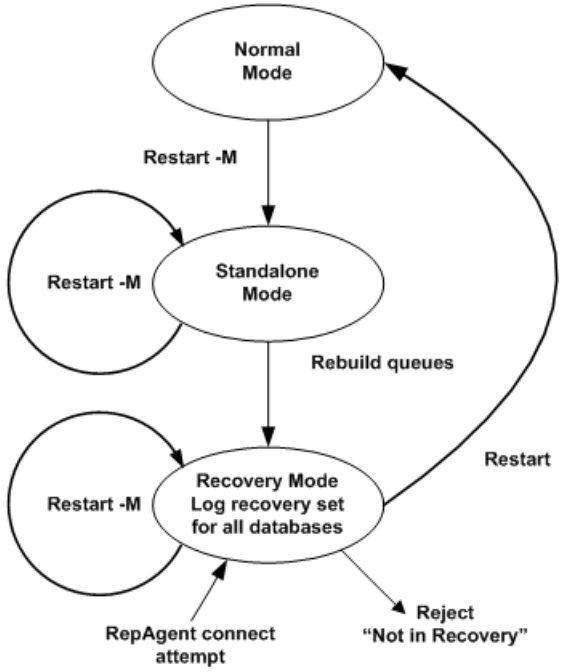
在恢复模式下，Replication Server 只允许处于恢复模式的 RepAgent 进行连接。如果 RepAgent 未处于恢复模式而尝试进行连接，Replication Server 会拒绝该连接，并显示以下错误消息：

```
Rep Agent not in recovery mode
```

如果您使用自动重新启动 RepAgent 并将其连接到 Replication Server 的脚本，则必须使用 **for_recovery** 选项启动 RepAgent。在正常模式下不允许 RepAgent 进行连接。

此图说明了使用 **rebuild queues** 命令从正常模式到独立模式再到恢复模式的过程。

图 27： 使用 rebuild queues 命令进入恢复模式



另请参见

- Replication Server 独立模式 （第 313 页）

Replication Server 独立模式

独立模式允许您查看稳定队列的内容，因为既没有消息写入队列，也不从队列中读取消息。

要在独立模式下启动 Replication Server，请使用 **-M** 标志。独立模式有助于查看 Replication Server 的状态，因为状态是静态的。

独立模式在以下方面与 Replication Server 的正常模式有所不同：

- 不接受传入连接。如果任何 RepAgent 或 Replication Server 尝试在独立模式下连接到 Replication Server，将会引发“`Replication Server 处于独立模式下`”消息。
- 不启动传出连接。独立模式下的 Replication Server 不会尝试连接到其它 Replication Server。
- 不启动 DSI 线程，即使 DSI 队列中有尚未应用的消息。
- 不启动“分配器”(DIST) 线程。DIST 线程从入站队列读取消息，执行预订分析，然后将消息写入出站队列。

重建稳定队列后进行丢失检测

Replication Server 执行丢失检测以确定重建稳定队列后是否有任何无法恢复的消息。

通过检查 Replication Server 丢失检测消息，您可以确定需要何种用户干预（如果有）才能将所有数据恢复到系统中。

Replication Server 会在重建稳定队列后检测两种类型的丢失：

- SQM 丢失 – 在两个 Replication Server 之间丢失的数据，会在下一个下游节点检测到这类丢失
- DSI 丢失 – 在 Replication Server 和它管理的复制数据库之间丢失的数据

如果所有数据都可用，则没有必要进行干预，复制系统可以返回到正常操作。例如，如果您知道为路由或连接设置的保存间隔比故障时间长，您无需进行任何干预就可以恢复所有消息。但是，如果未设置保存间隔或者设置的间隔太小，则可能会丢失某些消息。

注意： 检测到丢失的 Replication Server 不会接受来自源的消息。丢失检测会防止源截断其稳定队列。

例如，如果 Replication Server RS2 检测到复制数据服务器 DS2.RDB 丢失了来自主数据服务器 DS1.PDB 的数据，则在您确定处理丢失的方式之前，Replication Server RS1 无法截断其队列。其结果是，RS1 可能会用尽稳定存储空间。在检测到丢失之前（即在报告“正在检测丢失”消息后），您可以为成对的源和目标选择忽略丢失。

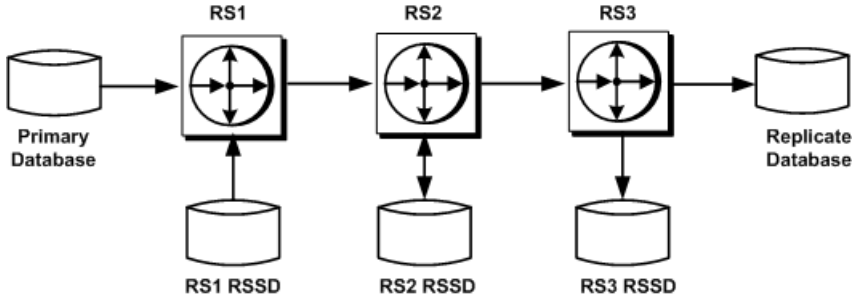
两个 Replication Server 之间的 SQM 丢失

了解 Replication Server 如何检测两个 Replication Server 之间的数据丢失。

每次您在恢复过程中重建稳定队列时，Replication Server 都会从发送其分配数据的节点请求积压消息。如果 Replication Server 管理主数据库，它会指示主数据库的 RepAgent 从联机事务日志的开始发送消息。积压消息将会重新填充清空的稳定队列。

Replication Server 会在您要重建的那些具有源自 Replication Server 的直接路由的节点上启用丢失检测模式。在此图中，如果重建 Replication Server RS2 的队列，Replication Server RS3 将会检测丢失。同样，如果您重建 Replication Server RS1 的队列，RS2 将会检测丢失。

图 28： 复制系统丢失检测示例



当您在 RS2 上执行 **rebuild queues** 命令时，RS3 会对所有主数据库执行丢失检测，这些主数据库的更新会通过 RS2 路由到 RS3。RS3 会记录每个数据库的消息。如果您在 RS3 上重建队列，将不会执行任何 SQM 丢失检测，因为没有源自 RS3 的路由。

Replication Server 通过查看重复的消息来检测丢失。如果 RS3 接收到在执行 **rebuild queues** 命令之前所接收到的消息，则说明未丢失消息。如果 RS3 在执行 **rebuild queues** 后接收到的第一条消息在以前未见到过，则说明有消息丢失，或稳定队列中以前没有任何消息。

即使稳定队列中没有来自特定源的消息，RS3 也会将其标识为丢失，因为没有重复消息可用于比较。您可以通过创建间隔小于保存间隔的心跳来防止这类假丢失检测。这样，可以确保稳定队列中至少总有一条消息。

SQM 示例

当 RS3 对重建的 RS2 执行了 SQM 丢失检测后，它会在其日志文件消息中记录类似以下“正在检测丢失”消息示例的消息。这些消息标志着丢失检测过程的开始。后续的消息将记录结果。每条消息都包含一对源和目标。

第一条示例消息表示 RS3 正在检查 RS3 的 RSSD 与 RS2 的 RSSD 之间是否有丢失：

```
Checking Loss for DS3.RS3_RSSD from DS2.RS2_RSSD
date=Nov-01-95 10:15 am
qid=0x01234567890123456789
```

第二条示例消息表示 RS3 正在检测 RS3 的复制数据库 `RDB` 与 RS1 的主数据库 `PDB` 之间是否有丢失：

```
Checking Loss for DS3.RDB from DS1.PDB
date=Nov-01-95 11:00am
qid=0x01234567890123456789
```

第三条示例消息表示 RS3 正在检查 RS3 的 RSSD 与 RS1 的 RSSD 之间是否有丢失：

```
Checking Loss for DS3.RS3_RSSD from DS1.RS1_RSSD
date=Nov-01-95 10:00am
qid=0x01234567890123456789
```

RS3 会报告它是否检测到丢失。例如，这种丢失检测测试的结果可能显示如下：

```
No Loss for DS3.RS3_RSSD from DS2.RS2_RSSD
```

```
Loss Detected for DS3.RDB from DS1.PDB
```

```
No Loss for DS3.RS3_RSSD from DS1.RS1_RSSD
```

Replication Server 与其数据库之间的 DSI 丢失

了解 Replication Server 如何检测 Replication Server 和它管理的复制数据库之间的数据丢失。

Replication Server 队列中的某些消息的目标为数据库，而不是其它 Replication Server。DSI 以一种类似于稳定队列丢失检测的方式执行丢失检测。

如果您在没有起始路由的 Replication Server 上重建队列，则不会执行 SQM 丢失检测，但 Replication Server 会对其消息执行 DSI 丢失检测。

DSI 示例

Replication Server RS2 中的 DSI 为 RS2 中的 RSSD 生成以下消息：

```
DSI: detecting loss for database DS2.RS2_RSSD from origin
DS1.RS1_RSSD
date=Nov-01-95 10:58pm
qid=0x01234567890123456789
```

当保留的消息开始从先前节点到达时，DSI 会检测到丢失，这取决于 DSI 是否已经看到了来自源的第一条消息。如果它未检测到丢失，则会生成一条类似以下内容的消息：

```
DSI: no loss for database DS2.RS2_RSSD from origin DS1.RS1_RSSD
```

如果 DSI 检测到丢失，则会生成一条类似以下内容的消息：

```
DSI: loss detected for database DS2.RS2_RSSD from origin DS1.RS1_RSSD
```

处理丢失

当 Replication Server 检测到丢失后，SQM 或 DSI 的连接上将不再接受任何消息。

例如，当 RS3 检测到 `RDB` 数据库与 `PDB` 数据库之间有 SQM 消息丢失时，它会拒绝所有从 `PDB` 数据库到 `RDB` 数据库的后续消息。

恢复丢失

要恢复丢失，您需要选择三个选项之一。

您可以选择：

- 忽略丢失并继续，即使可能会丢失某些消息。可以使用包含 **rs_subcmp** 程序的预订比较过程，可以在该程序中使用 **-r** 标志调和主数据和复制数据。

另请参见《Replication Server 管理指南第一卷》中的“管理预订”和《Replication Server 参考手册》的“可执行程序”中的“**rs_subcmp**”。

- 忽略丢失，然后删除并重新创建预订。
- 通过从脱机日志重放事务来恢复（只适用于主 Replication Server 丢失）。在这种情况下，您将不忽略丢失。

另请参见

- 使用预订比较过程 （第 302 页）

恢复丢失

在某些情况下，您必须执行 **ignore loss** 命令。

执行 **ignore loss**：

- 如果您选择通过重新创建预订或重放日志来恢复丢失消息。
- 对于 SQM 丢失，在报告丢失的 Replication Server 上执行，以强制该 Replication Server 再次开始接受消息。例如，要在 Replication Server 中忽略 RS3 从 DS1.PDB 中检测到的丢失，请在 RS3 上输入以下命令：

  ```
  ignore loss from DS1.PDB to DS3.RDB
  ```
- 对于 DSI 丢失，在检测到丢失的 Replication Server 数据库上执行。例如，要忽略 DS2.RS2_RSSD 中报告的源自源 DS1.RS1_RSSD 的丢失，请在 RS2 上输入以下命令：

  ```
  ignore loss from DS1.RS1_RSSD to DS2.RS2_RSSD
  ```
- 当您相继重建两个 Replication Server 时，位于路由目标的 Replication Server 同时检测到 SQM 和 DSI 丢失。

  在这种情况下，您需要执行两次 **ignore loss**，一次用于 SQM 丢失，一次用于 DSI 丢失。在目标 Replication Server 上执行的用于忽略 DSI 丢失的 **ignore loss** 命令与用于忽略源自上一节点的 SQM 丢失的命令相同。

为数据库设置日志恢复

手工设置日志恢复是从截断的主数据库日志进行脱机恢复或从转储恢复主数据库和复制数据库的一部分。

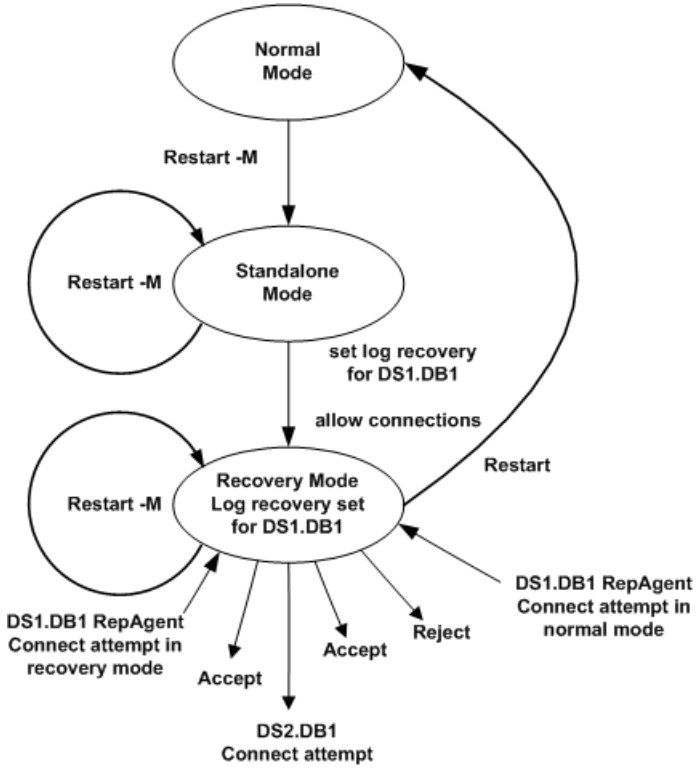
虽然在脱机重建队列的过程中会自动为所有数据库设置日志恢复，但是通过手工设置日志恢复，您可以在无需重新构建稳定队列的情况下恢复每个数据库。

set log recovery 命令可将 Replication Server 置于数据库的日志恢复模式。您要在将 Replication Server 置于独立模式后执行此命令。要将 RepAgent 只连接到那些已设置了日志恢复模式的数据库，请执行 **allow connections** 命令。这会将 Replication Server 置于恢复模式。

此图说明了使用 **set log recovery** 和 **allow connections** 命令从正常模式到独立模式再到恢复模式的进程。

对于使用 **set log recovery** 命令指定的数据库，Replication Server 只接受来自处于恢复模式的其它 Replication Server 和 RepAgent 的连接。然后，您要将事务转储恢复到临时恢复数据库中。

图 29：使用 allow connections 命令进入恢复模式



设置日志恢复后进行丢失检测

在将临时恢复数据库应用到主数据库时，Replication Server 可能会检测主数据库和管理该主数据库的 Replication Server 之间的 SQM 丢失。

如果所有数据都可用，则没有必要进行干预，复制系统可以返回到正常操作。Replication Server 会记录如下消息：

```
No Loss Detected for DS1.PDB from DS1.PDB
```

如果没有足够的消息，Replication Server 会记录一条类似以下内容的丢失检测消息：

```
Loss Detected for DS1.PDB from DS1.PDB
```

您必须确定是通过执行 **ignore loss** 命令忽略丢失还是从头开始重复恢复过程。要忽略丢失，请在主 Replication Server 上输入以下命令：

```
ignore loss for DS1.PDB from DS1.PDB
```

如果您接收到丢失检测消息，则说明未能将数据库重装到足以检索所有消息的旧有状态。您必须正确确定要装载的转储。

另请参见

- 确定要装载的转储 （第 318 页）

确定要装载的转储

装载事务日志转储时，要始终检查丢失检测期间显示的“正在检测丢失”消息。

如果有多条消息，请选择最早的日期和时间来确定要装载哪些转储。

例如，如果 Replication Server 生成以下消息，请装载就在 1995 年 11 月 1 日 22:58 之前进行的转储：

```
Checking Loss for DS3.RDB from DS1.PDB
date=Nov-01-2011 10:58pm
qid=0x01234567890123456789
```

该消息中的 `date` 是原始队列生成 Replication Server 所接收到的最后一条消息时，日志中时间最久的打开事务的日期和时间。找到 `timestamp` 早于该消息中的日期和时间的最近一个事务转储。然后，找到在该事务转储之前进行的完整数据库转储。

原始队列 ID（或 `qid`）由 RepAgent 形成，用于标识事务日志中的日志记录。`date` 作为 `timestamp` 嵌入 `qid` 中。Replication Server 可以将 `timestamp` 转换为日期以便用于 Adaptive Server 的 RepAgent。

非 Sybase 数据服务器的复制代理也可以将 `timestamp` 嵌入 `qid` 中。Replication Server 将针对非 Sybase 数据服务器转换 `timestamp`（20 – 27 字节）。是否使用这些字节取决于 Replication Agent。

注意： 如果数据服务器不是 Adaptive Server，则消息中显示的日期可能为无效日期。您可能需要对 20 – 27 字节的 `qid` 进行解码，才能确定要装载的转储。

调整数据库生成号

每次装载数据库进行恢复时，可能都会需要更改数据库生成号，如您要使用的恢复过程中所示。

复制系统中的每个主数据库都包含一个数据库生成号。此号码同时存储在数据库和管理该数据库的 Replication Server 的 RSSD 中。

数据库生成号的最大值是 65,535。除非有绝对的必要，否则 Sybase 建议您避免将生成号增加到较高的值。

如果要重置数据库生成号，必须重建复制环境。重建环境包括删除并重新创建到要重置数据库生成号的主数据库的连接以及重建主数据库的复制配置。

确定数据库生成号
了解何时调整数据库生成号。

主数据库的 RepAgent 将数据库生成号放置在它为传递给 Replication Server 的每个日志记录所构建的 `qid` 的最高位的 2 字节中。

`qid` 的其余部分由其它信息构建，这些信息提供记录在日志中的位置，并确保 `qid` 随每个传递给 Replication Server 的记录递增。

`qid` 值递增这一要求允许 Replication Server 检测重复记录。例如，当重新启动 RepAgent 时，它可能会重新发送某些 Replication Server 已经处理过的日志记录。如果 Replication Server 接收到一个记录，该记录的 `qid` 比它处理的上一个记录小，它会将该记录视为重复记录，并忽略该记录。

如果您要将主数据库恢复到以前的状态，请增加数据库生成号，这样 Replication Server 就不会忽略在重装数据库后提交的日志记录。只有在从转储中装载主数据库或从协调的转储中装载时，此步骤才适用。

如果您要重放日志记录，则只有在 RepAgent 先前使用更大的生成号发送重装日志记录的情况下，才应增加数据库生成号。只有在因为发生第一次故障而必须恢复数据库并登录到以前的状态，而后，又因为发生第二次故障而必须重放日志时，才会出现这种情况。

警告！ 只有在恢复过程中才可更改数据库生成号。在其它任何时候更改该号码都可能会导致复制数据库出现重复数据或丢失数据。

另请参见
- 从转储装载主数据库 （第 296 页）
- 从协调的转储装载 （第 297 页）

转储和数据库生成号
了解在重新装载转储后如何以及何时调整数据库生成号。

重装数据库转储时，数据库生成号将包括在恢复的数据库中。由于数据库生成号也存储在管理该数据库的 Replication Server 的 RSSD 中，所以，您可能需要更新该号码，以便使其与恢复的数据库中的生成号相匹配。

但是，当您重装事务日志时，数据库生成号不包括在恢复的日志中。例如，假定数据库中发生了以下操作：

表 34. 转储和数据库生成号

运算	数据库生成号
数据库转储 D1	100
事务转储 T1	100

运算	数据库生成号
dbcc settrunc('ltm', 'gen_id', 101)	101
事务转储 T2	101
数据库转储 D2	101

如果您重装数据库转储 D1，则数据库生成号 100 将随之恢复。如果您重装事务转储 T1，则生成号保留为 100。进行事务转储 T2 后，生成号仍为 100，因为重装事务转储不会更改数据库生成号。在这种情况下，在让 RepAgent 扫描事务转储 T2 之前，您需要使用 **dbcc settrunc** 命令将生成号更改为 101。

但是，如果您在重新开始复制前装载数据库转储 D2，则无需更改数据库生成号，因为生成号 101 已经恢复。

重置主数据库生成号

了解如何重置数据库生成号。

在此过程中，主数据库是指要重置数据库生成号的主数据库。

1. 在复制 Replication Server 中，删除所有引用为到主数据库的连接定义的复制定义和发布的预订。
2. 删除在步骤 1 中删除的预订引用的所有发布。
3. 删除在步骤 2 中删除的发布引用的所有项目。
4. 在主 Replication Server 中，删除主数据库连接的所有复制定义。
5. 在主 Replication Server 中，删除到主数据库的连接以及到预订主数据库的复制数据库的所有连接。
6. 在主数据库上将数据库生成号设置为 0：
   - 在 Adaptive Server 中：

     ```
     dbcc settrunc(‘ltm’, ‘gen_id’, 0)
     ```
   - 在 UNIX 和 Windows 上的 Replication Agent for IBM DB2 UDB、Microsoft SQL Server 和 Oracle 中：

     ```
     pdb_gen_id 0
     ```
7. 在主 Replication Server 中，创建到主数据库的新连接，然后创建到复制数据库的连接。
8. 重新创建删除的所有复制定义、发布、项目和预订。请参见《Replication Server 管理指南第一卷》的“使用 Sybase Central 管理复制环境”中的“设置复制环境”。

非 ASE 数据库支持

您可以为充当主数据库的所有支持的非 ASE 数据库重置数据库生成号。

有关支持的主数据库，请参见《Replication Server 异构复制指南》。

Adaptive Server 的复制数据库重新同步

Replication Server 允许重新同步和实现复制数据库，并继续进一步的复制，而不会丢失数据或存在数据不一致的风险，且不会强制停顿主数据库。

数据库重新同步基于从受信任的源获取数据转储并将该转储应用于要重新同步的目标数据库。

若要重新同步 Oracle 数据库，请参见《Replication Server 异构复制指南》中的“Oracle 复制数据库重新同步”。

配置数据库重新同步

可以使用 Replication Server 和 RepAgent 中的命令和参数来配置数据库重新同步。

1. 停止挂起的 RepAgent 进行的复制处理。
2. 使 Replication Server 处于 resync 模式。
   在 resync 模式下，Replication Server 跳过事务并清除复制队列中的复制数据，预期使用从主数据库或受信任的源获取的转储来重新填充复制数据库。
3. 重新启动 RepAgent 并向 Replication Server 发送 resync database 标记以指示正在进行重新同步。
4. 验证 DSI 是否收到 resync database 标记。
5. 从主数据库获取转储。

   当 Replication Server 检测到指示主数据库转储完成的 dump 标记时，Replication Server 会停止跳过事务，并且可以确定应用于复制数据库的事务。
6. 验证 DSI 是否收到 dump database 标记。

   注意： 发送 dump database 标记不适用于使用 **init** 指令发送 resync 标记的情况。
7. 将转储应用于复制数据库。
8. 重新开始复制。

指示 Replication Server 跳过事务

结合使用 **skip to resync** 参数和 **resume connection** 命令来指示 Replication Server 跳过指定复制数据库的 DSI 外发队列中的事务，直到 Replication Server 收到并确认 RepAgent 发送的 dump database 标记。

Replication Server 不处理外发队列中的记录，因为复制数据库中的数据将替换为转储内容。

请参见《Replication Server 参考手册》的“Replication Server 命令”中的“**resume connection**”。

运行：

```
resume connection to data_server.database skip to resync
marker
```

警告！ 如果在错误的连接上执行 **resume connection** 和 **skip to resync marker** 选项，则复制数据库上的数据将变得不同步。

在设置 **skip to resync marker** 后，Replication Server 不会在 Replication Server 日志或数据库例外日志中记录跳过的事务。Replication Server 在您设置 **skip [n] transaction** 后记录跳过的事务。

将 resync database 标记发送到 Replication Server

指示 RepAgent 向 Replication Server 发送 resync database 标记来指示正在进行重新同步工作。

当以 resync 模式重新启动 RepAgent 时，RepAgent 会向 Replication Server 发送 resync database 标记，作为发送任何 SQL 数据定义语言 (DDL) 或数据操作语言 (DML) 事务之前的第一个消息。同一主数据库的多个复制数据库均会收到相同的 resync 标记，因为它们都具有 DSI 外发队列。

对于使用 **skip to resync marker** 参数重新开始的每个 DSI，DSI 外发队列在 Replication Server 系统日志中记录 DSI 已收到 resync 标记，还记录从该点起，DSI 拒绝提交的事务，直到它收到 dump database 标记为止。

在 Adaptive Server 中，结合使用 **sp_start_rep_agent** 和 **resync**、**resync purge,** 或 **resync init** 参数来支持用于发送 resync database 标记的相应选项。

不使用任何选项发送 Resync 标记

当截断点没有更改时，可使用不带任何选项的 **sp_start_rep_agent** 发送 resync 标记，例外情况是 RepAgent 应从它处理的最后一个点继续处理事务日志。

语法： **sp_start_rep_agent** *database_name*, **'resync'**

每个外发 DSI 线程和队列接收并处理 resync database 标记。DSI 在收到 resync 标记时会报告给 Replication Server 系统日志，从而满足 DSI 的 skip to resync 标记请求。随后，DSI 在等待 dump database 标记时拒绝提交的事务。使用此消息和对等待 dump database 标记的行为的更改，可以将任何转储应用于复制数据库。

使用 purge 指令发送 Resync 标记

结合使用 **sp_start_rep_agent** 和 **purge** 选项来发送 resync 标记，以指示 Replication Server 在接收任何新入站事务之前，清除入站队列中所有打开的事务并重置重复检测。

语法： **sp_start_rep_agent** *database_name*, **'resync purge'**

在移动主数据库的截断点后将使用 **purge** 选项，此情况在您执行以下操作时发生：

- 手动更改截断点。
- 禁用 RepAgent。

- 执行 Adaptive Server 命令，例如 **dbcc dbrepair**。

由于截断点已更改，必须清除 Replication Server 入站队列中打开的事务，因为这些事务与从新辅助截断点发送的新活动不匹配。Replication Server 重置重复检查，因为更改的截断点可以使用以前的原始队列 ID (OQID) 发送记录。由于从队列中清除了以前的数据，Replication Server 不会将来自 RepAgent 的任何新活动视为重复活动，因此不会拒绝新活动。purge 选项没有更改 DSI 处理，因为 Replication Server 在等待 dump database 标记时继续拒绝外发队列命令。

使用 init 命令发送 Resync 标记

通过使用 **sp_start_rep_agent** 和 init 选项的 **init** 命令发送 resync 标记，以指示 Replication Server 清除入站队列中所有打开的事务、重置重复检测和挂起外发 DSI。

语法： **sp_start_rep_agent** *database_name*, **'resync init'**

使用此选项可通过与复制数据库相同的转储重新装载主数据库。由于没有从主数据库获取任何转储，RepAgent 没有发送 dump database 标记。在 resync 标记之后不再等待 dump database 标记，**init** 选项会在 Replication Server 处理 resync 标记后立即挂起 DSI 连接。

挂起 DSI 后，通过 DSI 的所有后续活动都将包括新事务。在通过用于主数据库的相同转储重新装载复制数据库后，可以重新开始 DSI。

获取数据库的转储

使用 **dump database** Adaptive Server 命令。

请参见《Adaptive Server Enterprise 系统管理指南第二卷》的“制订备份和恢复计划”中的“使用 **dump** 和 **load** 命令”。

将 dump database 标记发送到 Replication Server

在您获取主数据库的转储时，RepAgent 会自动生成 dump database 标记并将其发送给 Replication Server。

注意： 当您使用 **init** 指令发送 resync 标记时，不适合发送 dump database 标记。

在将转储应用于复制数据库后，可以手动恢复 DSI。将复制在 dump database 标记指示的转储点之后提交的事务。

监控 DSI 线程信息

使用 **admin who** 命令可提供有关数据库重新同步期间的 DSI 信息。

状态	说明
SkipUntil Resync	DSI 在您执行以下命令后重新开始：**skip to resync.** 此状态一直保持到 DSI 收到 resync database 标记为止。
SkipUntil Dump	DSI 已收到 resync database 标记。此状态一直保持到 DSI 处理了后续 dump database 标记为止。

将转储应用于要重新同步的数据库

仅当您在系统日志中看到相关消息后，才能将主数据库转储装载到复制数据库。

- 当 Replication Server 收到带或不带 **purge** 选项的 resync database 标记和 dump database 标记时：

```
DSI for data_server.database received and processed
Resync Database Marker. Waiting for Dump Marker.
```

```
DSI for data_server.database received and processed
Dump Marker. DSI is now suspended. Resume after database has been
reloaded.
```

- 当 Replication Server 收到带 **init** 标记的 resync database 时：

```
DSI for data_server.database received and processed
Resync Database Marker. DSI is now suspended. Resume after
database has been reloaded.
```

请参见《Adaptive Server Enterprise 参考手册：命令》中“命令”中的“**load database**”，以了解有关将转储装载到要重新同步的数据库中的说明。

数据库重新同步方案

在不同的方案中，按照相应的过程来重新同步数据库。在完成某个过程后，主数据库和复制数据库在事务上将保持一致。

若要执行某个过程，您必须：

- 是复制系统管理员
- 具有成功运行的现有复制环境
- 具有可用于将数据从主数据库复制到复制数据库的方法和进程

有关用于 Adaptive Server 和 Replication Server 的 RepAgent 的命令和语法，请参见和《Replication Server 管理指南第一卷》中的“管理 RepAgent 和支持 Adaptive Server”。

直接从主数据库重新同步一个或多个复制数据库

从单个主数据库重新同步一个或多个复制数据库。

此过程具有微小变动，允许您：

- 当主数据库和复制数据库之间的复制延迟导致无法使用复制恢复数据库，并且基于复制数据的报告由于延迟而无法使用时，重新填充复制数据库。
- 使用主数据库中的受信任数据重新填充复制数据库。
- 当主数据库是多个复制数据库的源时，协调重新同步。
- 如果主点是包含要与一个或多个复制数据库重新同步的数据库的热备份对的逻辑数据库，则协调重新同步。在热备份对中，活动数据库充当主数据库，备用数据库充当复制数据库。因此，主点上热备份对的活动数据库还显示为一个或多个复制数据库的主数据库。

直接从主数据库重新同步

从主数据库重新同步复制数据库。

1. 停止 RepAgent 进行的复制处理。在 Adaptive Server 中，执行：

   ```
   sp_stop_rep_agent database
   ```

2. 挂起 Replication Server DSI 与复制数据库的连接：

   ```
   suspend connection to dataserver.database
   ```

3. 指示 Replication Server 移除复制数据库外发队列中的数据，并等待来自主数据库 RepAgent 的 resync 标记：

   ```
   resume connection to data_server.database skip to
   resync marker
   ```

4. 指示 RepAgent 以 resync 模式启动并向 Replication Server 发送 resync 标记：
   - 如果截断点未从其原始位置移动，则在 Adaptive Server 中执行：

     ```
     sp_start_rep_agent database, 'resync'
     ```

   - 如果截断点已从其原始位置移动，则在 Adaptive Server 中执行：

     ```
     sp_start_rep_agent database, 'resync purge'
     ```

5. 在 Replication Server 系统日志中，通过查找以下消息来验证 DSI 是否已收到并接受来自 RepAgent 的 resync 标记：

   ```
   DSI for data_server.database received and processed
   Resync Database Marker. Waiting for Dump Marker.
   ```

   注意： 如果要重新同步多个数据库，请检验要重新同步的每个数据库的 DSI 连接是否已接受 resync 标记。

6. 获取主数据库内容的转储。请参见《Adaptive Server Enterprise 参考手册：命令》中“命令”中的“**dump database**”。Adaptive Server 自动生成 dump database 标记。

7. 通过在 Replication Server 系统日志中查找以下消息来验证 Replication Server 是否已处理 dump database 标记：

   ```
   DSI for data_server.database received and processed
   Dump Marker. DSI is now suspended. Resume after database has been
   reloaded.
   ```

   当 Replication Server 收到 dump 标记时，DSI 连接将自动挂起。

8. 将主数据库的转储应用于复制数据库。请参见《Adaptive Server Enterprise 参考手册：命令》中“命令”中的“**load database**”。

9. 在将转储应用于复制数据库后，重新开始 DSI：

   ```
   resume connection to data_server.database
   ```

使用第三方转储实用程序重新同步

在使用第三方转储实用程序（例如磁盘快照）转储主数据库后，协调重新同步。

第三方工具不像本地数据库转储实用程序那样与主数据库紧密交互。如果您的第三方工具没有在主数据库事务日志中记录 RepAgent 可用于生成 dump database 标记的任何内容，请生成您自己的 dump database 标记来完成重新同步过程。请参见您的第三方工具文档。

1. 停止 RepAgent 进行的复制处理。在 Adaptive Server 中，执行：

   ```
   sp_stop_rep_agent database
   ```

2. 挂起 Replication Server DSI 与复制数据库的连接：

   ```
   suspend connection to dataserver.database
   ```

3. 指示 Replication Server 移除复制数据库外发队列中的数据，并等待来自主数据库 RepAgent 的 resync 标记：

   ```
   resume connection to data_server.database skip to
   resync marker
   ```

4. 使用第三方转储实用程序获取主数据库内容的转储。
5. 根据您获取转储时主数据库中的信息或第三方工具中的信息确定转储点。使用第三方工具时，您负责确定转储点。例如，如果您使用磁盘复制工具，则可以临时中断主数据库上的活动来从磁盘快照中消除正在进行的事务，然后使用“事务日志的结尾”点作为 dump database 标记。
6. 在主数据库上执行 **rs_marker** 存储过程，以便 RepAgent 标记您在步骤 5 中获取的转储位置的结尾：

   ```
   rs_marker “dump database database_name 'current date' oqid"
   ```

   其中，*current date* 是 `datetime` 格式的任意值，*oqid* 是任何有效的十六进制值。请参见《Replication Server 参考手册》的“主题”的“数据类型”的“日期/时间以及日期和时间数据类型”中的“日期/时间值的条目格式”。

   例如，可以使用日期和时间值“20110915 14:10:10”以及 *oqid* 值 0x0003 在 `rdb1` 数据库上标记转储位置的结尾：

   ```
   rs_marker “dump database rdb1 '20110915 14:10:10' 0x0003"
   ```

   RepAgent 自动为您在步骤 6 中标记的位置生成 dump database 标记，并将其发送到 Replication Server。

7. 指示 RepAgent 以 resync 模式启动并向 Replication Server 发送 resync 标记：
   - 如果截断点尚未从其原始位置移动，则在 Adaptive Server 中执行以下命令：

     ```
     sp_start_rep_agent database, 'resync'
     ```

   - 如果截断点已从其原始位置移动，则在 Adaptive Server 中执行以下命令：

     ```
     sp_start_rep_agent database, 'resync purge'
     ```

8. 通过在 Replication Server 系统日志中查找以下消息，来验证 DSI 是否已收到并接受来自 Replication Agent 的 resync 标记：

```
DSI for data_server.database received and processed
Resync Database Marker. Waiting for Dump Marker.
```

9. 通过在 Replication Server 系统日志中查找以下消息来验证 Replication Server 是否已处理 dump database 标记：

```
DSI for data_server.database received and processed
Dump Marker. DSI is now suspended. Resume after
database has been reloaded.
```

   当 Replication Server 收到 dump 标记时，DSI 连接将自动挂起。

10. 通过第三方工具将主数据库的转储应用于复制数据库。请参见您的 Adaptive Server 和第三方工具文档。
11. 在将转储应用于复制数据库后，重新开始 DSI：

```
resume connection to data_server.database
```

不支持 Resync Database 标记时的重新同步

如果 RepAgent 或主数据库尚未更新为支持自动生成 resync 标记，则协调重新同步。

注意： 只能将此过程用于 Adaptive Server。

1. 挂起 Replication Server DSI 与复制数据库的连接：

```
suspend connection to dataserver.database
```

2. 指示 Replication Server 移除复制数据库外发队列中的数据，并等待来自主数据库 RepAgent 的 resync 标记：

```
resume connection to data_server.database skip to
resync marker
```

3. 确保系统日志中没有任何打开的事务，然后在主数据库中手动生成 **resync 标记**：

```
execute rs_marker ‘resync database’
```

4. 在 Replication Server 系统日志中，通过查找以下消息来验证 DSI 是否已收到并接受来自 RepAgent 的 resync 标记：

```
DSI for data_server.database received and processed
Resync Database Marker. Waiting for Dump Marker.
```

5. 获取主数据库内容的转储。

   Adaptive Server 自动生成 dump database 标记。请参见。

6. 通过在 Replication Server 系统日志中查找以下消息来验证 Replication Server 是否已处理 dump database 标记：

```
DSI for data_server.database received and processed
Dump Marker. DSI is now suspended. Resume after database has been
reloaded.
```

当 Replication Server 收到 dump 标记时，DSI 连接将自动挂起。

7. 将主数据库的转储应用于复制数据库。请参见《Adaptive Server Enterprise 参考手册：命令》中“命令”中的“**load database**”。
8. 在将转储应用于复制数据库后，重新开始 DSI：

```
resume connection to data_server.database
```

通过相同转储重新同步主数据库和复制数据库

协调重新同步以通过相同的转储或数据副本来重新装载主数据库和复制数据库。无需任何 dump database 标记，因为您没有从主数据库获取转储。

1. 停止 RepAgent 进行的复制处理。不要改变截断点。

   在 Adaptive Server 中，执行：

   ```
   sp_stop_rep_agent database
   ```

2. 挂起 Replication Server DSI 与复制数据库的连接：

   ```
   suspend connection to data_server.database
   ```

3. 指示 Replication Server 移除复制数据库外发队列中的数据，并等待来自主数据库 RepAgent 的 resync 标记：

   ```
   resume connection to data_server.database skip to
   resync marker
   ```

4. 在应用转储之前获取 RepAgent 设置。

   注意： Adaptive Server 在数据库中存储 RepAgent 使用的连接设置和其它配置。如果您通过从不同数据库获取的转储装载主数据库，则 RepAgent 会丢失其配置设置，或者设置发生更改以与您从其获取转储的数据库的设置相匹配。

5. 将外部源中的数据转储应用于主数据库。

   应用转储后，将 RepAgent 配置重置为应用转储前存在的设置。

6. 确保最后一个主数据库事务日志页没有包含可通过在主 Adaptive Server 数据库上执行来影响复制数据库表的任何操作：

   ```
   rs_update_lastcommit 0, 0, 0, ""
   go 100
   ```

7. 将截断点移动到主数据库的事务日志的结尾。在 Adaptive Server 中，执行：

   ```
   dbcc settrunc('ltm', 'end')
   go
   ```

8. 使用 **init** 指令指示 RepAgent 以 resync 模式启动。在 Adaptive Server 中，执行：

   ```
   sp_start_rep_agent database, 'resync init'
   ```

9. 通过在 Replication Server 系统日志中查找以下消息，来验证 DSI 是否已收到并接受来自 RepAgent 的 resync 标记：

```
DSI for data_server.database received and processed
Resync Database Marker. DSI is now suspended. Resume
after database has been reloaded.
```

当 Replication Server 使用 **init** 标记接收并处理 resync database 后，DSI 连接将挂起。

10. 将外部源中的数据转储应用于复制数据库。
11. 在将转储应用于复制数据库后，重新开始复制数据库的 DSI，以允许 Replication Server 应用主数据库中的事务：

```
resume connection to data_server.database
```

重新同步热备份应用程序中的活动数据库和备用数据库

当热备份对是单个主数据库的复制节点时，可重新同步热备份环境中的活动数据库和备用数据库。

在此方案中，复制节点是包括充当单个逻辑数据库的活动数据库和备用数据库的热备份对。

主 ---> Replication ---> 复制逻辑数据库

数据库 Server [活动+备用热备份对]

重新同步方案过程是一个包含两个步骤的过程：使用主数据库中的转储重新同步热备份对的复制活动数据库，然后使用活动数据库中的转储或主数据库中的现有转储重新同步热备份对的复制备用数据库。

1. 停止主数据库 RepAgent 和热备份活动数据库 RepAgent 进行的复制处理。

   在 Adaptive Server 中，执行：

   ```
   sp_stop_rep_agent database
   ```
2. 挂起 Replication Server DSI 与活动数据库和备用数据库的连接：

   ```
   suspend connection to dataserver.database
   ```
3. 指示 Replication Server 移除活动数据库和备用数据库的外发队列中的数据，并等待来自主数据库 RepAgent 的 resync 标记：

   ```
   resume connection to data_server.database skip to
   resync marker
   ```
4. 指示主数据库 RepAgent 以 resync 模式启动并向 Replication Server 发送 resync 标记。
   - 如果截断点尚未从其原始位置移动，则在 Adaptive Server 中执行以下命令：

     ```
     sp_start_rep_agent database, 'resync'
     ```
   - 如果截断点已从其原始位置移动，则在 Adaptive Server 中执行以下命令：

     ```
     sp_start_rep_agent database, 'resync purge'
     ```

5. 通过在 Replication Server 系统日志中查找以下消息，来验证活动数据库的 DSI 是否已收到并接受来自主数据库 RepAgent 的 resync 标记：

   ```
   DSI for data_server.database received and processed
   Resync Database Marker. Waiting for Dump Marker.
   ```

6. 获取主数据库内容的转储。请参见《Adaptive Server Enterprise 参考手册：命令》中“命令”中的“**dump database**”。Adaptive Server 自动生成 dump database 标记。

7. 在应用转储之前获取 RepAgent 设置。

   注意： Adaptive Server 在数据库中存储 RepAgent 使用的连接设置和其它配置。如果您通过从不同数据库获取的转储装载主数据库，则 RepAgent 会丢失其配置设置，或者设置发生更改以与您从其获取转储的数据库的设置相匹配。

8. 通过在 Replication Server 系统日志中查找活动数据库的以下消息，来验证活动数据库的 Replication Server DSI 是否已处理 dump database 标记：

   ```
   DSI for data_server.database received and processed
   Dump Marker. DSI is now suspended. Resume after database has been
   reloaded.
   ```

9. 将主数据库的转储应用于活动数据库。请参见《Adaptive Server Enterprise 参考手册：命令》中“命令”中的“**load database**”。

   应用转储后，将 RepAgent 配置重置为应用转储前存在的设置。

10. 确保最后一个主数据库事务日志页没有包含可通过在主 Adaptive Server 数据库上执行来影响复制数据库表的任何操作：

    ```
    rs_update_lastcommit 0, 0, 0, ""
    go 100
    ```

11. 将截断点移动到活动数据库的事务日志的结尾。在 Adaptive Server 中，执行：

    ```
    dbcc settrunc('ltm', 'end')
    go
    ```

12. 使用 **init** 指令指示 RepAgent 以 resync 模式启动。在 Adaptive Server 中，执行：

    ```
    sp_start_rep_agent database, 'resync init'
    ```

13. 通过在 Replication Server 系统日志中查找以下消息，来验证备用数据库的 DSI 是否已收到并接受来自活动数据库 RepAgent 的 resync 标记：

    ```
    DSI for data_server.database received and processed
    Resync Database Marker. DSI is now suspended. Resume
    after database has been reloaded.
    ```

    当 Replication Server 使用 **init** 标记接收并处理 resync database 后，DSI 连接将挂起。

14. 获取活动数据库内容的转储并将该转储应用于备用数据库。如果转储没有包含数据库配置信息，您也可以应用步骤 6 中的主数据库的转储。

15. 重新开始活动数据库和备用数据库的 DSI：

    ```
    resume connection to data_server.database
    ```

异步过程

了解异步存储过程以及复制与表复制定义关联的存储过程的方法。各种要求存储过程的应用程序均支持此方法。

有关与函数复制定义关联的复制存储过程的信息，请参见《Replication Server 管理指南第一卷》中的“管理复制函数”。

请参见《Replication Server 设计指南》以了解与复制的存储过程有关的复制系统设计问题的信息。

异步过程传递简介

通过异步过程传递，您可以执行用于在主数据库或复制数据库中进行复制操作的 SQL 存储过程。

由于使用了 **sp_setreplicate** 或 **sp_setrepproc** 系统过程将这些存储过程标记为要复制，因此它们也称为复制的存储过程。

为了满足分布式应用程序的要求，Replication Server 提供了两种类型的异步存储过程传递：应用存储过程和请求存储过程。

Adaptive Server 记录的复制存储过程

了解 Adaptive Server 如何确定将在哪个数据库中记录复制存储过程的执行情况。

存储过程将记录在启动封装事务的数据库中。

- 如果用户没有显式开始执行事务，则在存储过程执行之前，Adaptive Server 会在用户的当前数据库中开始一个事务。
- 如果用户在某个数据库中开始执行事务，然后在另一个数据库中执行复制的存储过程，则执行仍将被记录到用户开始执行事务的数据库中。

如果表样式复制存储过程（使用 **sp_setreplicate***proc_name*, **'true'** 或 **sp_setrepproc***proc_name*, **'table'** 标记为要复制）的执行情况记录在一个数据库中，并在另一个数据库中更改复制表，则表的更改情况和过程执行情况分别记录在不同的数据库中。因此，执行存储过程的结果可能被复制两次：第一次是复制存储过程执行本身；第二次是复制在另一个数据库中记录的表更改。

记录复制存储过程的限制

复制的 Adaptive Server 存储过程不能包含带有 `text`、`unitext` 或 `image` 数据类型的参数。

请参见《Adaptive Server 参考手册》。

混合型事务
如果调用一个或多个请求存储过程的事务是一个混合型事务，而且它还执行应用存储过程或包含数据修改语言，则 Replication Server 将在完成其它所有操作后再处理这些请求存储过程。

所有请求操作都在一个单独的事务中一起进行处理。如果单个 Replication Server 同时管理主数据和复制数据，则可能会出现这种情况。

应用存储过程

Replication Server 从主数据库向复制数据库传递的复制的存储过程称为应用存储过程。

您可以使用应用存储过程传递将首先在主数据上执行的事务复制到复制数据库中。数据更改会先应用到主数据库，然后再分配给预订数据复制定义的复制数据库。Replication Server 将作为维护用户在复制数据库中执行复制的存储过程，这与普通的数据复制是一致的。

您可以使用应用存储过程来实现重要的性能优势。例如，如果您的组织有大量的行更改，就可以创建一个可更改许多行的应用存储过程，而不用逐个复制每一行。还可以使用应用存储过程来复制数据集更改，这种更改很难用普通的预订来实现。有关详细信息，请参见《Replication Server 设计指南》。

通过用存储过程的第一个语句更新某个表，您可以设置应用存储过程。还必须确保目标数据库具有对更新的行的操作前映像和操作后映像的预订。应用存储过程必须只更新复制的表中的一行。Replication Server 使用存储过程更新过的第一个行来确定将该过程的用户定义的函数发送到什么位置。

如果不符合设置应用存储过程的规则，Replication Server 就无法向复制数据库分配存储过程。当 Replication Server 无法传递应用存储过程时，将会出现一些警告情况并采取相应的操作。

另请参见
- 警告情况 （第 336 页）

请求存储过程

Replication Server 从复制数据库向主数据库传递的复制的存储过程称为请求存储过程。

您可以使用请求存储过程将事务从复制数据库传递回主数据库。

例如，位于远程位置的客户端应用程序可能需要更改主数据。这种情况下，远程位置的应用程序就可以在本地执行请求存储过程，以更改主数据。Replication Server 通过

在复制数据库中执行与主数据库中的存储过程同名的存储过程，将此请求存储过程传递到主数据库。主数据库中的存储过程即会更新事务更改的主数据。

Replication Server 会以在复制数据库中执行存储过程的用户身份在主数据库中执行复制的存储过程。这样就可以确保只有授权的用户才可以更改主数据。

在实际应用中，Replication Server 可以复制主数据库中更改的部分或是全部数据。这些更改通过对相关数据的预订，以数据行（插入、删除或更新操作）或存储过程的方式传播到由 Replication Server 管理的复制数据库中。利用这种机制，事务结果可以迅速到达主数据库和复制数据库。

警告！ 请不要在主数据库中执行请求存储过程。这可能会引发循环现象，在这种情况下，复制 Replication Server 会造成同一过程在主数据库中反复执行。

使用请求存储过程可确保所有更新都在主数据库中进行，从而保留 Replication Server 的基本主副本数据模型，同时还可避免网络故障和过大的流量影响复制系统。即使主数据库出现故障，或者从复制数据库到主数据库之间出现网络故障，Replication Server 仍然具有容错功能。它会对所有未传递的请求存储过程调用进行排队，直到发生故障的组件重新恢复联机状态。组件可以重新工作后，Replication Server 就会完成传递。

通过利用 Replication Server 的有保障的请求存储过程传递功能，您可以获得使用一个确定的、包含所有最新更改的数据副本的所有优点。同时，Replication Server 还可以取消复制数据库的应用程序与主数据库的连接，从而提高了可用性和改善了性能。

有关与异步过程传递相关的复制系统设计问题的详细信息，请参见《Replication Server 设计指南》。

异步存储过程的先决条件

在实现应用存储过程或请求存储过程时，应满足一些先决条件。

- 了解如何利用异步存储过程来满足实际应用需要。请参见《Replication Server 设计指南》。
- 设置存储过程的 RepAgent，即使数据库中不包含任何主数据（例如在使用请求函数时）。请参见适用于您的平台的 Replication Server 安装和配置指南。
- 对于 Replication Server 没有生成缺省函数字符串的函数字符串类，请为该函数字符串类创建一个用户定义的函数的函数字符串。可以使用 **alter function string** 命令将缺省函数字符串替换为一个执行应用程序要求的操作的函数字符串。

注意： 对于提供了缺省生成的函数字符串的函数字符串类，Replication Server 将创建一个缺省的函数字符串，它可以执行与用户定义的函数同名的存储过程。本节中的任务假定 Replication Server 处理的是针对这样一些类的应用存储过程或请求存储过程。对于所有其它类，必须为用户定义的函数字符串创建函数字符串。

另请参见

- 函数字符串和函数字符串类 （第 27 页）

实现应用存储过程

了解用于实现应用存储过程的步骤。

前提条件

验证是否满足异步存储过程的前提条件。

过程

有关用来查询 RSSD 中是否存在系统信息的存储过程的信息，请参见《Replication Server 参考手册》中的“RSSD 存储过程”。

1. 设置含有复制表的复制数据库。这些表可能符合主表的复制定义，也可能不符合。
2. 根据需要，设置从主 Replication Server 到复制 Replication Server 的路由，复制 Replication Server 具有对主表的复制定义的预订。

   请参见《Replication Server 管理指南第一卷》中的“管理路由”。
3. 在主 Replication Server 上找到或创建用于标识要修改的表的复制定义。

   请参见《Replication Server 管理指南第一卷》中的“管理复制表”。
4. 在主数据库中，使用 **sp_setreplicate** 或 **sp_setreptable** 系统过程将表标记为要复制。

   例如，对于名为 `employee` 的表，请输入以下命令之一：
   - `sp_setreplicate employee, 'true'`

     在将存储过程和表指定为要复制时，请遵循相应的准则。
   - `sp_setreptable employee, 'true'`

     对于 **sp_setreptable**，单引号是可选的。请参见《Replication Server 管理指南第一卷》的“管理复制表”的“将表标记为要复制”中的“使用 sp_setreptable 系统过程”。
5. 在主数据库上创建存储过程。

   存储过程的第一个语句必须包含用于主表第一行的更新命令。例如：

```
create proc upd_emp
 @emp_id int, @salary float
 as
 update employee
 set salary = salary * @salary
 where emp_id = @emp_id
```

警告！ 如果存储过程的第一个语句包含的操作不是 **update**，则 Replication Server 无法将存储过程分配给复制数据库。检查警告情况。请不要在存储过程中包含 **dump**

transaction 或 **dump database** 命令。如果存储过程包含的命令有语句级错误，则这种错误可能出现在复制 DSI 上。根据错误操作的不同，DSI 可能会关闭。

6. 在主数据库中，使用 **sp_setreplicate** 或 **sp_setrepproc** 系统过程将存储过程标记为要复制。

   例如，输入以下命令之一：

   - `sp_setreplicate upd_emp, 'true'`

     在将存储过程和表指定为要复制时，请遵循相应的准则。
   - `sp_setrepproc upd_emp, 'table'`

     请参见《Replication Server 管理指南第一卷》的“管理复制函数”中的“将存储过程标记为要复制”。
7. 在复制 Replication Server 中，针对主数据库的存储过程更新的表，创建对复制定义的预订。

   请参见《Replication Server 管理指南第一卷》中的“管理预订”。

   警告！ 请确保复制数据库对所更新行的操作前映像和操作后映像都进行预订。如果不是这样，Replication Server 就无法将存储过程分配给复制数据库。
8. 使用与主数据库上的存储过程相同的名称和参数在复制数据库上创建存储过程，但不要将该存储过程标记为已复制。

   例如：

   ```
   create proc upd_emp
    @emp_id int, @salary float
    as
    update employee
    set salary = salary * @salary
    where emp_id = @emp_id
   ```
9. 为维护用户授予存储过程的 **execute** 权限。

   例如：

   ```
   grant execute on upd_emp to maint_user
   ```
10. 在主 Replication Server 上创建用户定义的函数，该函数将存储过程与它更新的表的复制定义的名称相关联。

    例如：

    ```
    create function employee_rep.upd_emp
     (@emp_id int, @salary float)
    ```

    同一个表的所有复制定义只共享一个用户定义的函数。您可以为这些复制定义中的任何一个指定名称。
11. 验证所有步骤中的所有 Replication Server 和数据库对象在相应的位置中均已存在。

另请参见

- 异步存储过程的先决条件 （第 333 页）
- 将存储过程和表指定为要复制 （第 339 页）

警告情况

如果应用存储过程未传送到复制数据库，则会出现 Replication Server 警告情况。

如果应用存储过程的第一个语句不是更新操作，或者复制数据库没有预订更新的行的操作前映像或操作后映像，则 Replication Server 无法将应用存储过程传递到复制数据库。相反，Replication Server 会执行其它操作，您可以将这些操作视为警告。

Replication Server 执行的操作取决于:

- 在主数据库的应用存储过程中包含的第一个操作（非更新）
- 行修改是否保留在复制数据库的预订中，以及它是否与预订的前映像或后映像相匹配

警告情况和 Replication Server 操作

- 条件：第一项行操作是插入操作。
  操作：Replication Server 将分配插入操作而不是应用存储过程。
- 条件：第一项行操作是删除操作。
  操作：Replication Server 将分配删除操作而不是应用存储过程。
- 条件：Replicate Replication Server 具有与所修改行的操作前映像（而不是操作后映像）相匹配的预订。
  操作：Replication Server 会将一个删除操作（**rs_delete** 系统函数）分配给复制数据库，并预订行修改的操作前映像，但不预订操作后映像。
  示例：假定有一个表 `T1`，该表有一个名为 `C1` 的列，其值为 1。复制数据库有一个对表 `T1` 的复制定义的预订，其中 `C1` = 1。
  如果使用参数 = 1（操作前映像）和参数 = 2（操作后映像）来执行相关存储过程，则复制数据库不会预订操作后映像的值 2。因此，Replication Server 会将删除操作分配给复制数据库。
- 条件：Replicate Replication Server 具有与所修改行的操作后映像（而不是操作前映像）相匹配的预订。
  操作：Replication Server 会将一个插入操作（**rs_insert** 系统函数）分配到复制数据库，并预订行修改的操作后映像，但不会预订操作前映像。
  示例：假定有一个表 `T1`，该表有一个名为 `C1` 的列，其值为 1。复制数据库有一个对表 `T1` 的复制定义的预订，其中 `C1` = 2。
  如果使用参数 = 1（操作前映像）和参数 = 2（操作后映像）来执行相关存储过程，则复制数据库不会预订操作前映像的值 1。因此，Replication Server 会将插入操作分配到复制数据库。
- 条件：Replicate Replication Server 具有既不与所修改行的操作前映射匹配也不与所修改行的操作后映像匹配的预订。
  操作：Replication Server 不会将任何操作或存储过程分配给复制数据库。
  示例：假定有一个表 `T1`，该表有一个名为 `C1` 的列，其值为 1。复制数据库有一个对表 `T1` 的复制定义的预订，其中 `C1` > 2。

如果使用等于 1（操作前映像）和等于 2（操作后映像）的参数来执行关联的存储过程，复制 Replication Server 就不会预订操作前映像值 1 或操作后映像值 2。因此，Replication Server 不会对复制数据库执行分配操作。

实现请求存储过程

了解用于实现请求存储过程的步骤。

前提条件

验证是否满足异步存储过程的前提条件。

过程

有关用来查询 RSSD 中是否存在系统信息的存储过程的信息，请参见《Replication Server 参考手册》中的“RSSD 存储过程”。

1. 根据需要，设置从复制 Replication Server 到主 Replication Server（在其中更新数据）的路由以及从主 Replication Server 到发送更新的复制 Replication Server 的路由。

   请参见《Replication Server 管理指南第一卷》中的“管理路由”。
2. 在主 Replication Server 上，为复制 Replication Server 上的用户创建登录名和口令。

   请参见《Replication Server 管理指南第一卷》中的“管理 Replication Server 安全性”。
3. 在复制 Replication Server 上，为该用户创建在主 Replication Server 上执行存储过程所需的权限。

   请参见《Replication Server 管理指南第一卷》中的“管理 Replication Server 安全性”。
4. 在主 Replication Server 上，找到或创建用来标识要修改的表的复制定义。

   有关创建复制定义的信息，请参见《Replication Server 管理指南第一卷》中的“管理复制表”。

   复制 Replication Server 可以具有复制定义的预订。
5. 在复制数据库上创建存储过程，该过程不执行任何更新。

   例如：

   ```
   create proc upd_emp
    @emp_id int, @salary float
    as
    print "Transaction accepted."
   ```

   如果希望该存储过程的名称与其它复制数据库中的存储过程的名称相同，请遵循为用户定义的函数指定非唯一名称的准则。

6. 在复制数据库中，使用 **sp_setreplicate** 或 **sp_setrepproc** 系统过程将存储过程标记为要复制。

   例如，输入以下命令之一：

   ```
   sp_setreplicate upd_emp, 'true'
   ```

   在将存储过程和表指定为要复制时，请遵循相应的准则。

   或者

   ```
   sp_setrepproc upd_emp, 'table'
   ```

   请参见《Replication Server 管理指南第一卷》的“管理复制函数”中的“将存储过程标记为要复制”。
7. 在主数据库上创建存储过程，使用与复制数据库中的存储过程相同的名称，但不要将该过程标记为已复制。此存储过程可修改主表。

   例如：

   ```
   create proc upd_emp
    @emp_id int, @salary float
    as
    update employee
    set salary = salary * @salary
    where emp_id = @emp_id
   ```

   注意： 如果您为了使用不同名称来执行存储过程而更改了函数的函数字符串，则主数据库和复制数据库上的存储过程名称就可能不一致。

   您可以将函数映射到不同的存储过程名称。
8. 将存储过程的权限授予将执行此存储过程的复制 Replication Server 用户。

   例如：

   ```
   grant all on upd_emp to public
   ```
9. 在主 Replication Server 上创建用户定义的函数，该函数将存储过程与它更新的表的复制定义的名称相关联。

   例如：

   ```
   create function employee_rep.upd_emp
    (@emp_id int, @salary float)
   ```
10. 验证所有步骤中的所有 Replication Server 和数据库对象在相应的位置中均已存在。

另请参见

- 异步存储过程的先决条件 （第 333 页）
- 为用户定义的函数指定非唯一名称 （第 343 页）
- 将存储过程和表指定为要复制 （第 339 页）
- 将函数映射到不同的存储过程名称 （第 342 页）

将存储过程和表指定为要复制

您可以使用 Adaptive Server 中的 **sp_setreplicate** 系统过程将数据库表和存储过程标记为要复制。

您还可以使用 **sp_setreptable** 系统过程将表标记为要复制，使用 **sp_setrepproc** 系统过程将存储过程标记为要复制。这些系统过程扩展了 **sp_setreplicate** 的功能，并且将会取代它。

sp_setreplicate 系统过程的语法是：

```
sp_setreplicate [object_name [, {' true' | 'false' ]]
```

object_name 可以是表名或存储过程名。

“true”和“false”参数将更改指定对象的复制状态。（单引号内是可选内容。）

- 使用不带参数的 **sp_setreplicate** 可列出数据库中所有复制的对象。
- 使用只带对象名的 **sp_setreplicate** 可检查该对象的复制状态。如果已经为该对象已启用了复制，Adaptive Server 将报告 **'true'**，反之则报告 **'false'**。
- 使用带有对象名和 **'true'** 或 **'false'** 的 **sp_setreplicate**，可启用或禁用该对象的复制。您必须是 Adaptive Server 系统管理员或数据库所有者，才能使用 **sp_setreplicate** 更改对象的复制状态。

警告！ 复制的存储过程只应修改它所执行的数据库中的数据。如果它修改了另一个数据库中的数据，Replication Server 将会复制更新的数据和存储过程。

管理用户定义的函数

了解用于管理用户定义的函数的命令。

可通过更改用户定义的函数的函数字符串来自定义数据库操作以及显示函数相关信息。

有关使用这些命令所需的权限列表，另请参见《Replication Server 管理指南第一卷》中的“管理 Replication Server 安全性”。

另请参见

- 自定义数据库操作 （第 11 页）

创建用户定义的函数

可以使用 **create function** 命令在 Replication Server 上注册复制的存储过程。

当存储过程执行时，Replication Server 会将它映射到某个复制定义。复制定义包含一个用户定义的函数名称，该名称与存储过程名称匹配。

Replication Server 将函数传递到主要用于复制定义的 Replication Server。拥有复制定义的目标 Replication Server 接收到函数后，会将存储过程的参数映射到用户定义函数的命令中。

create function 命令的语法是：

```
create function replication_definition.function
([@parameter datatype [, @parameter datatype]...])
```

replication_definition 必须是现有的复制定义。

使用此命令时，请遵循以下准则：

- 请在已创建复制定义的 Replication Server 上执行此命令。
- 不要使用保留的系统函数名称。
- 包括所列参数外的小括号，即使所定义的函数不带参数。
- 如果没有使用提供了缺省生成函数字符串的函数字符串类，请在创建用户定义的函数后，使用 **create function string** 命令来添加函数字符串。

以下示例创建一个名为 **Stock_receipt** 的用户定义的函数。该函数与 `Items_rd` 复制定义相关联：

```
create function Items_rd.Stock_receipt
 (@Location int, @Recpt_num int,
 @Item_no char(15), @Qty_recd int)
```

在用户执行此复制的存储过程时，Replication Server 会立即将其传递出去。

另请参见

- 创建函数字符串 （第 33 页）
- 系统函数摘要 （第 14 页）

将参数添加到用户定义的函数中

可以使用 **alter function** 命令通知 Replication Server 在复制的存储过程中添加的新参数。

1. 在主数据服务器或复制数据服务器上更改存储过程，并提供新参数的缺省值。
2. 作为预防措施，请停顿系统。在更新过程中更改函数可能会产生无法预知的后果。

   请参见《Replication Server 管理指南第一卷》的“管理复制系统”中的“停顿 Replication Server”。
3. 使用 **alter function** 命令更改函数。
4. 如果没有使用提供了缺省生成函数字符串的函数字符串类，请更改函数字符串以使用新参数。

alter function 命令的语法是：

```
alter function replication_definition.function
add parameters @parameter datatype
[, @parameter datatype]...
```

replication_definition 是函数的复制定义的名称。一个函数最多可以有 255 个参数。

以下示例将一个名为 Volume 的 int 参数添加到复制定义为 Tokyo_quotes 的 **New_issue** 函数中：

```
alter function Tokyo_quotes.New_issue
 add parameters @Volume int
```

另请参见

- 更改函数字符串 （第 36 页）

删除用户定义的函数

可以使用 **drop function** 命令删除用户定义的函数。

此命令可删除函数名和为函数创建的所有函数字符串。不能删除系统函数。

在删除用户定义的函数前，请确保：

1. 在主数据库中使用 **drop procedure** Adaptive Server 命令删除存储过程。
   （可选）使用 **sp_setreplicate** 或 **sp_setrepproc** 系统过程并指定 **'false'** 以便为存储过程禁用复制。
   在将存储过程和表指定为要复制时，请遵循相应的准则。有关使用 **sp_setrepproc** 的详细信息，请参见《Replication Server 管理指南第一卷》的“管理复制函数”中的“将存储过程标记为要复制”。
2. 作为预防措施，请在执行 **drop function** 命令前请先停顿系统。在更新过程中删除函数可能会产生无法预知的后果。
   请参见《Replication Server 管理指南第一卷》的“管理复制系统”中的“停顿 Replication Server”。

drop function 命令的语法是：

```
drop function replication_definition.function
```

请在创建复制定义的 Replication Server 上执行该命令。

以下命令删除以前创建的 **Stock_receipt** 用户定义的函数：

```
drop function Items_rd.Stock_receipt
```

另请参见

- 将存储过程和表指定为要复制 （第 339 页）

将函数映射到不同的存储过程名称

了解如何将用户定义的函数映射到不同的存储过程名称。

如果您在某个数据库中创建了用户定义的函数，且该函数使用提供了缺省生成的函数字符串的函数字符串类，则 Replication Server 会生成一个缺省的函数字符串。缺省生成的函数字符串执行与用户定义的函数具有相同名称和参数的存储过程。

例如，如果您正使用缺省函数字符串，通过在主数据库中创建一个与用户定义的函数具有相同名称和参数的存储过程，您可以设置要在复制数据库中执行的请求存储过程。

如果您希望将用户定义的函数映射到其它存储过程名，请使用 **alter function string** 命令对 Replication Server 进行配置，使其通过执行其它名称的存储过程来传递存储过程。您也可以在允许自定义函数字符串的函数字符串类中执行此操作。

示例

本示例说明了如何将用户定义的函数映射到不同的存储过程名称。

1. 假定主 Adaptive Server 上存在存储过程 **upd_sales**，而且该过程对 Adaptive Server `sales` 表进行了一次更新：

   ```
   create proc upd_sales
    @stor_id varchar(10),
    @ord_num varchar(10),
    @date datetime
    as
   64 update sales set date = @date
    where stor_id = @stor_id
    and ord_num = @ord_num
   ```

2. 要在 Replication Server 上注册 **upd_sales** 存储过程，请创建以下函数，其名称包括两个部分：`sales` 表中的 `sales_def` 复制定义和 **upd_sales** 复制的存储过程：

   ```
   create function sales_def.upd_sales
    (@stor_id varchar(10), @date datetime)
   ```

3. 在复制 Adaptive Server 上，创建一个同名但不执行任何操作的存储过程 **upd_sales**：

   ```
   create proc upd_sales
    @stor_id varchar(10),
    @ord_num varchar(10),
    @date datetime
    as
    print "Attempting to Update Sales Table"
    print "Processing Update Asynchronously"
   ```

4. 要执行带有名称 **real_update** 而不是 **upd_sales** 的 **upd_sales** 存储过程，请执行以下步骤：
   - 更改缺省生成的函数字符串：

     ```
     alter function string sales_def.upd_sales
      for rs_sqlserver_function_class
      output rpc
      'execute real_update
     ```

```
@stor_id = ?stor_id!param?,
@date = ?date!param?'
```

- 在主数据库中创建一个名为 **real_update** 的存储过程。它可接受两个参数。

为用户定义的函数指定非唯一名称

用户定义的函数名在复制系统全局范围内必须是唯一的，这样 Replication Server 才能找到定义用户定义的函数的特定复制定义。

如果对同一个主表创建了多个复制定义，则只有一个用户定义的函数可以用于该表的所有复制定义。

如果用户定义的函数名不是唯一的，则存储过程的第一个参数必须是 *@rs_repdef*，并且在执行存储过程时，复制定义的名称必须用该参数进行传递。

对于用户定义的函数，请不要在 **create function** 命令中定义 *@rs_repdef* 参数。Replication Agent 会提取复制定义名，然后用 LTL 命令发送它。这种约定适用于 RepAgent for Adaptive Server，但其它数据服务器的 Replication Agent 可能不支持这种约定。

示例

本示例假定用户定义的函数不是唯一的，并且在执行以下存储过程时，复制定义名被传递给 *@rs_repdef* 参数：

```
create proc upd_sales
 @rs_repdef varchar(255),
 @stor_id varchar(10),
 @date datetime
 as
 print "Attempting to Update Sales Table"
 print "Processing Update Asynchronously"
```

Sun Cluster 2.2 的高可用性

了解在 Sun Cluster 2.2 上配置 Sybase Replication Server 以实现高可用性所需的背景知识和操作步骤。

Sybase Replication for Sun Cluster HA 简介

如果要使用 Sybase Replication for Sun Cluster HA，应满足一些假设条件。

这些假设包括：

- 您熟悉 Sybase Replication Server。
- 您熟悉 Sun Cluster HA。
- 您具有装有 Sun Cluster HA 2.2 的双节点集群硬件系统。

参考文档：

- 《Sun Cluster 2.2 软件规划和安装指南》
- 《Sun Cluster 2.2 系统管理指南》
- 《配置 Sybase Adaptive Server Enterprise 12.0 Server 以获得高可用性：Sun Cluster HA》（请参见白皮书）

术语

了解 Sybase Replication Server for Sun Cluster HA 中讨论的术语。

使用的术语包括：

- 集群 - 多个系统或节点，作为一个实体协同工作，向用户提供应用程序、系统资源和数据。
- 集群节点 – 组成 Sun Cluster 的物理机。也称为物理主机。
- 数据服务 – 一种应用程序，在网络上提供客户端服务，并实现对基于磁盘的数据的读写访问。例如，Replication Server 和 Adaptive Server Enterprise 就是数据服务。
- 磁盘组 – 一组明确定义的多主机磁盘，可作为一个单元在 HA 配置中的两台服务器之间移动。
- 故障监控器 – 一种可以探查数据服务的守护程序。
- 高可用性 (HA) – 即极短的停机时间。提供 HA 的计算机系统通常提供 99.999% 的可用性，或大约每年 5 分钟的意外停机时间。
- 逻辑主机 – 一组资源，包括磁盘组、逻辑主机名和逻辑 IP 地址。逻辑主机驻留在集群机的某个物理主机（或节点）上，或者受物理主机（或节点）控制。它可作为一个单元在集群中的物理主机之间移动。

- Master – 对逻辑地址映射到它的 Ethernet 地址的磁盘组具有排它读写访问权的节点。逻辑主机的当前 Master 运行逻辑主机的数据服务。
- 多主机磁盘 – 配置为可从多个节点进行访问的磁盘。
- 故障切换 – 由节点故障或数据服务故障触发的事件，发生此事件时，逻辑主机和逻辑主机上的数据服务会移到另一个节点。
- 故障恢复 – 一个预先计划好的事件，发生此事件时，逻辑主机及其上的数据服务会移回原始主机。

技术概述

Sun Cluster HA 是基于硬件和软件的解决方案，它在集群机上提供高可用性支持，并且可在几秒钟内自动进行数据服务故障切换。它通过增加硬件冗余、软件监控和重新启动的功能来实现此功能。

Sun Cluster 为系统管理员进行配置、维护 HA 安装和排除 HA 安装故障提供了集群管理工具。

Sun Cluster 配置具备针对以下单点故障的容错能力：

- 服务器硬件故障
- 磁盘介质故障
- 网络接口故障
- 服务器操作系统故障

发生以上的任何一种故障时，HA 软件可将逻辑主机故障切换到其它节点，并在新节点的逻辑主机上重新启动数据服务。

Sybase Replication Server 是作为集群计算机的逻辑主机上的数据服务实现的。Replication Server 的 HA 故障监控器定期探查 Replication Server。如果 Replication Server 已停止响应，故障监控器会尝试在本地重新启动 Replication Server。如果 Replication Server 在可配置的时间段内再次失败，故障监控器将故障切换到逻辑主机，以便 Replication Server 在另一节点上重新启动。

对于 Replication Server 客户端而言，这种情形就如同原始 Replication Server 重新启动一样。但逻辑主机已移到另一个物理机的事实对用户来说则是透明的。Replication Server 附属于逻辑主机，而不是物理机。

作为一项数据服务，Replication Server 包含在 Sun Cluster 中注册为回调方法的一组脚本。Sun Cluster 会在故障切换的不同阶段调用这些方法：

- FM_STOP - 关闭要进行故障切换的数据服务的故障监控器。
- STOP_NET – 关闭数据服务本身。
- START_NET – 在新节点上启动数据服务。
- FM_START - 在新节点上启动数据服务的故障监控器。

每台 Replication Server 都使用 **hareg** 命令注册为数据服务。如果集群上运行着多台 Replication Server，您必须注册每台 Replication Server。每项数据服务都具有自己的作为单独进程的故障监控器。

注意： 有关 **hareg** 命令的详细信息，请参见相应的 Sun Cluster 文档。

配置 Replication Server 以获得高可用性

了解在 Sun Cluster（假定是一台双节点集群机）上配置 Replication Server 以获得 HA 所需的任务。

系统应具备以下组件：

- 在 CPU、内存等资源方面具有相似配置的两台同构 Sun Enterprise 服务器。应使用集群互连来配置服务器，集群互连是用于维护集群可用性、同步和完整性的。
- 系统应配备一组多主机磁盘。多磁盘主机可为高可用 Replication Server 存储数据(分区）。只有当一个节点是磁盘所属逻辑主机的当前主节点时，该节点才可以访问多主机磁盘上的数据。
- 系统应安装具有自动故障切换功能的 Sun Cluster HA 软件。多主机磁盘应在整个系统范围内具有唯一路径名。
- 为提供磁盘故障保护，应使用磁盘镜像（不是由 Sybase 提供）。
- 应配置逻辑主机。Replication Server 在逻辑主机上运行。
- 确保 Replication Server 的逻辑主机的多主机磁盘组中具有足够的磁盘空间供分区使用，并确保逻辑主机的任何潜在 Master 具有足够的内存供 Replication Server 使用。

安装 Replication Server 以获得 HA

在 Replication Server 安装过程中，除了适用于您的平台的 Replication Server 安装指南中所述的任务外，您还需要执行一些任务。

1. 作为 Sybase 用户，在共享磁盘或本地磁盘上装载 Replication Server。

   如果将它装载到共享磁盘上，则不能同时从两台机器访问此版本。如果是装载在本地磁盘上，请确保两台机器具有相同的版本路径。如果版本路径不同，请使用一个符号链接，以使其相同。

   例如，如果 node1 上的版本路径为 `/node1/repserver`，而在 node2 上的版本路径为 `/node2/repserver`，请将这两个版本路径链接到两个节点上的 `/repserver`，以使 *$SYBASE* 环境变量在整个系统范围内保持一致。

2. 将 Replication Server、RSSD 服务器和主数据服务器/复制数据服务器的条目添加到两台机器上的 `$SYBASE` 目录的 interfaces 文件中。

   在 interfaces 文件中使用 Replication Server 的逻辑主机名。

   注意： 要使用 LDAP 目录服务而不是 interfaces 文件，请在 Replication Server 配置文件的 DIRECTORY 部分中提供多个条目。如果与第一个条目连接失败，目录

控制层 (DCL) 会尝试与第二个条目连接，依此类推。如果不能与 DIRECTORY 部分中的任何条目建立连接，Open Client/Server 不会使用缺省 interfaces 文件尝试连接。

有关设置 LDAP 目录服务的信息，请参见适用于您的平台的配置指南。

3. 启动 RSSD 服务器。
4. 按照适用于您的平台的安装指南，在当前作为逻辑主机中的 Master 的节点上安装 Replication Server。请确保：
   a) 设置环境变量 SYBASE、SYBASE_REP 和 SYBASE_OCS
      例如，输入：
      ```
      setenv SYBASE /REPSERVER1210
      setenv SYBASE_REP REP-12_1
      setenv SYBASE_OCS OCS-12_0
      ```
      /REPSERVER1210 为版本目录。
   b) 为 Replication Server 选择运行目录，该目录中将包含 Replication Server 运行文件、配置文件和日志文件。
      运行目录应在两个节点上都存在，并且在两个节点上的路径完全相同（必要时可链接该路径）。
   c) 为 Replication Server 分区选择多主机磁盘。
   d) 从 `run` 目录中启动 **rs_init** 命令。
      输入：
      ```
      cd RUN_DIRECTORY
      $SYBASE/$SYBASE_REP/install/rs_init
      ```
5. 确保 Replication Server 已启动。
6. 作为 Sybase 用户使用相同路径将运行文件和配置文件复制的其它节点。在第二个节点上编辑运行文件，确保其包含配置文件和日志文件的正确路径（尤其是使用链接时）。

注意： 运行文件名必须为 `RUN_repserver_name`，其中的 *repserver_name* 是 Replication Server 的名称。您可以定义配置文件名和日志文件名。

安装作为数据服务的 Replication Server

您还需要执行一些专门任务以将 Replication Server 作为数据服务进行安装。

1. 作为 root 在两个集群节点上创建目录 `/opt/SUNWcluster/ha/repserver_name`，其中的 *repserver_name* 是您的 Replication Server 的名称。

   每台 Replication Server 都必须有它自己的目录，并在路径中包含服务器名。将以下脚本从 Replication Server 安装目录 `$SYBASE/$SYBASE_REP/sample/ha` 复制到两个集群节点上的以下路径：

   `/opt/SUNWcluster/ha/repserver_name`

   其中的 *repserver_name* 是您的 Replication Server 的名称：

```
repserver_start_net
repserver_stop_net
repserver_fm_start
repserver_fm_stop
repserver_fm
repserver_shutdown
repserver_notify_admin
```

如果这些脚本作为另一 Replication Server 数据服务的一部分已经存在于本地机器，则您可以创建以下目录作为指向脚本目录的链接：

```
/opt/SUNWcluster/ha/repserver_name
```

2. 作为 root 在两个节点上创建目录 `/var/opt/repserver`（如果该目录不存在）。
3. 作为 root 在两个节点上为要作为 Sun Cluster 数据服务安装的每台 Replication Server 创建文件 `/var/opt/repserver/repserver_name`，其中的 *repserver_name* 是您的 Replication Server 的名称。

   此文件应只包含以下形式的两行，不含空格，并且只能由 root 读取：

   ```
   repserver:logicalHost:RunFile:releaseDir:SYBASE_OCS:SYBASE_REP
   ```

   ```
   probeCycle:probeTimeout:restartDelay:login/password
   ```

   其中：

   - *repserver* – Replication Server 名称。
   - *logicalHost* – 运行 Replication Server 的逻辑主机。
   - *RunFile* – 运行文件的完整路径。
   - *releaseDir* – $SYBASE 安装目录。
   - *SYBASE_OCS* – 连接库所在的 $SYBASE 子目录
   - *SYBASE_REP* – Replication Server 所在的 $SYBASE 子目录。
   - *probeCycle* – 故障监控器两次探查起始时间之间的秒数。
   - *probeTimeout* – 以秒为单位的时间，经过该时间之后，故障监控器将中止正在运行的 Replication Server 探查，并设置超时条件。
   - *restartDelay* – Replication Server 两次重新启动之间的最短时间（以秒为单位）。如果在 Replication Server 重新启动之后 *restartDelay* 秒内故障监控器再次检测到需要重新启动的条件，它就会触发切换到另一主机的操作。这样就解决了重新启动数据库不能解决的情况。
   - *login/password* – 故障监控器用于 ping Replication Server 的登录/口令。

   将 Replication Server 作为数据服务安装之后，如果要更改探查的 *probeCycle*、*probeTimeout*、*restartDelay* 或 login/password，请将 SIGINT(2) 发送到监控器进程 (repserver_fm) 以刷新内存。

   ```
   kill -2 monitor_process_id
   ```

4. 作为 root 在两个节点上创建文件 `/var/opt/repserver/repserver_name.mail`，其中的 *repserver_name* 是您的 Replication Server 的名称。

此文件列出 Replication Server 管理员的 UNIX 登录名。登录名应该全部在一行中，用一个空格分隔。

如果故障监控器遇到任何需要干预的问题，它将按此列表发送邮件。

5. 将 Replication Server 作为 Sun Cluster 上的数据服务注册。

```
hareg -r repserver_name \
-b "/opt/SUNWcluster/ha/repserver_name" \
-m START_NET="/opt/SUNWcluster/ha/repserver_name/
repserver_start_net" \
-t START_NET=60 \
-m STOP_NET="/opt/SUNWcluster/ha/repserver_name/
repserver_stop_net" \
-t STOP_NET=60 \
-m FM_START="/opt/SUNWcluster/ha/repserver_name/
repserver_fm_start" \
-t FM_START=60 \
-m FM_STOP="/opt/SUNWcluster/ha/repserver_name/repserver_fm_stop"
\
-t FM_STOP=60 \
[-d sybase] -h logical_host
```

其中 *-d sybase* 为必需部分（如果 RSSD 在同一集群的 HA 下），*repserver_name* 为您的 Replication Server 的名称，并且必须在脚本的路径中。

6. 启用数据服务

输入：**hareg -y** *repserver_name*

将 Replication Server 作为数据服务进行管理

了解如何将 Replication Server 作为数据服务启动和关闭，并了解用于监控和故障排除的有用日志。

启动和关闭数据服务

了解在将 Replication Server 注册为数据服务后启动和关闭 Replication Server 的命令。

要启动 Replication Server（如果尚未运行）并且还为 Replication Server 启动故障监控器，请输入：

```
hareg -y repserver_name
```

要关闭 Replication Server，请输入：

```
hareg -n repserver_name
```

如果通过任何其它方式关闭或停止（注销）Replication Server，则故障监控器会重新启动该 Replication Server 或对其进行故障切换。

Sun Cluster for HA 日志

可以使用一些日志进行调试。

可以使用:

- Replication Server 日志 – Replication Server 在此处记录消息。使用此日志可查找 Replication Server 的信息性消息和错误消息。此日志位于 Replication Server `Run` 目录中。
- 脚本日志 – 数据服务 START 和 STOP 脚本在此处记录消息。使用此日志可查找运行这些脚本所产生的信息性消息和错误消息。此日志位于 `/var/opt/repserver/harep.log`。
- 主控台日志 – 操作系统在此处记录消息。使用此日志可查找硬件的信息性信息和错误信息。此日志位于 `/var/adm/messages`。
- CCD 日志 – 集群配置数据库（Sun Cluster 配置的一部分）在此处记录消息。使用此日志可查找 Sun Cluster 配置和运行情况的信息性消息和错误消息。此日志位于 `/var/opt/SUNWcluster/ccd/ccd.log`。

实现参考复制环境

您可以使用环境中提供的产品快速设置 Adaptive Server 到 Adaptive Server 或 Oracle 到 Oracle 参考复制环境。

参考复制环境实现

Replication Server 包括一个用于使用您的环境中的可用产品快速设置 Adaptive Server 到 Adaptive Server 或 Oracle 到 Oracle 复制的参考实现的工具集。

可以实现一个复制环境以说明 Replication Server 特性和功能。使用此工具集可:

1. 构建一个包含 Replication Server 以及主数据库和复制数据库的参考环境。
2. 配置复制环境。
3. 在主数据库上执行简单事务，并通过数据库级复制来复制更改。
4. 收集统计信息并监控步骤 3 中的复制处理中的计数器。
5. 清除参考复制环境。

参考实现工具集包括位于 `$SYBASE/refimp` 中的脚本。

注意: 参考实现构建的复制环境中包含单个 Replication Server、主数据库服务器和复制数据库服务器。您无法为多个复制系统组件配置参考环境拓扑。

平台支持

可以在 Replication Server 支持的所有平台上实现参考环境，Linux on IBM p-Series (Linux on Power) 64 位除外。必须使用 Cygwin 运行参考实现脚本，以便在 Replication Server 支持的任何 Microsoft Windows 平台上设置参考环境。

请访问 Cygwin 网站：http://www.cygwin.com。

参考实现组件

您必须具有支持的复制环境组件版本，然后才能实现参考环境。

Adaptive Server

可以使用支持的 Replication Server 和 Adaptive Server 版本构建 Adaptive Server 到 Adaptive Server 复制的参考实现环境。

表 35. Adaptive Server 参考实现的受支持的产品组件版本

Replication Server	Adaptive Server
15.5	15.0.3, 15.5

例如，可以使用 Replication Server 15.5 和 Adaptive Server 15.0.3 或 15.5 版构建 Adaptive Server 参考环境。

Oracle
还可以使用支持的 Replication Server、Oracle、Replication Agent for Oracle 和 ECDA Option for Oracle 版本构建 Oracle 到 Oracle 复制的参考实现环境。

表 36. Oracle 参考实现的受支持产品组件版本

Replication Server	Oracle	Replication Agent for Oracle	ECDA Option for Oracle
15.5	10.2	15.2	15.0 ESD #3

例如，可以使用 Replication Server 15.5、Oracle 10.2, Replication Agent 15.2 和 ECDA Option for Oracle 15.0 ESD #3 构建 Oracle 的参考实现环境。

参考环境的前提条件

在构建参考环境之前，您必须了解一些先决条件和信息。

1. 对于 Oracle，请验证您是否在 Oracle 版本目录中具有 **execute** 权限。例如，验证您是否可以手动创建实例。
2. 验证 Replication Server 或 Adaptive Server 版本目录下面的 `SYBASE.sh` 文件中的环境变量设置是否正确。如果无法验证设置是否正确，请删除或重命名该文件。
3. 验证是否可以在 bash shell 中使用 UNIX **grep**、**kill**、**awk** 和 **ps** 命令。

参考实现过程使用 Replication Server 版本目录中的 `interfaces` 文件。如果该文件在运行参考实现过程之前就已存在，该过程将通过增加文件扩展名来备份现有的文件。

对于 Oracle，参考实现过程重命名现有的 `tnsname.ora` 和 `listener.ora` 文件，然后为 Oracle 参考实现创建新的文件。

构建参考环境

执行 **buildenv** 脚本以自动创建 Replication Server、主数据服务器和复制数据服务器以及数据库。

输入：

```
buildenv -f config_file
```

可以使用 *config_file* 指定构建配置文件的名称和位置，您可以在该文件中指定一些参数。

如果成功执行 **buildenv**，则会看到：

```
Environment setup successfully completed.
```

参考实现配置文件

Sybase 在支持的 UNIX 和 Microsoft Windows 平台上为 Adaptive Server 到 Adaptive Server 和 Oracle 到 Oracle 复制提供了配置文件模板，可以使用这些模板为您的环境创建配置文件。

该文件位于 `$SYBASE/REP-15_5/REFIMP-01_0` 中。

表 37. 参考实现配置文件

主复制数据服务器到复制数据服务器和平台	配置文件
UNIX 上的 ASE 到 ASE	`ase_unix_refimp.cfg`
Windows 上的 ASE 到 ASE	`ase_win_refimp.cfg`
UNIX 上的 Oracle 到 Oracle	`ora_unix_refimp.cfg`
Windows 上的 Oracle 到 Oracle	`ora_win_refimp.cfg`

ase_unix_refimp.cfg 模板文件示例

根据您的环境提供目录位置和主机名等值。

```
#####################################################################
#######
# --- Part 1. release directory of repserver/ase/oracle/refimp ----#
#####################################################################
#######
#
# --- PLATFORM('unix': UNIX/Linux platform, 'win': Windows) ---#
#
os_platform=unix
# --- DATABASE ('ase': Adaptive Server Enterprise, 'ora': ORACLE) ---
#
#
db_type=ase
#
# --- RS RELEASE DIRECTORY --- #
#
rs release directory=/remote/repeng4/users/xiel/repserver
#
# --- RS RELEASE SUBDIRECTORY --- #
#
rs_sub_directory=REP-15_2
#
# --- ASE RELEASE DIRECTORY --- #
#
ase_release=/remote/repeng4/users/xiel/ase
#
# --- ASE/ORACLE RELEASE SUBDIRECTORY --- #
#
ase_subdir=ASE-15_0
```

```
#
# --- REFERENCE IMPLEMENTATION RELEASE DIRECTORY --- #
#
refimp_release_dir=/calm/repl/svr/refimp
#
#
#
# --- REFERENCE IMPLEMENTATION WORK DIRECTORY ---
#
refimp_work_dir=/remote/repeng4/users/xiel/test
#
# --- OCS RELEASE DIRECTORY --- #
#
ocs_release_directory=OCS-15_0
#
# --- PDS DEVICE NAME WITH FULL PATH --- #
#
pds_device_file=/remote/repeng4/users/xiel/pds
#
# --- RDS DEVICE NAME WITH FULL PATH --- #
#
rds_device_file=/remote/repeng4/users/xiel/rds
#
# --- rs_init RELEASE DIRECTORY --- #
#
rsinit_release=/remote/repeng4/users/xiel/repserver
#
#
# --- interface FILE NAME ---
#
ini_filename=interfaces
#
# --- HOST NAME ---
#
host_name=newgarlic
########################################################################
#######
# --- Part 2. login information of replication server and data server
---#
########################################################################
#######
#
# --- RS NAME --- #
#
rs_name=SAMPLE_RS
#
# --- RS USER NAME --- #
#
rs_username=sa
#
# --- RS PASSWORD --- #
#
rs_password=
#
#
#
```

```
# --- ERSSD NAME --- #
#
rssd_name=SAMPLE_RS_ERSSD
#
# --- ERSSD USER NAME --- #
#
rssd_username=rssd
#
# --- ERSSD PASSWORD --- #
#
rssd_password=rssd_pwd
#
# --- PDS NAME --- #
#
primary_ds=PDS
#
# --- PDB NAME ---
#primary_db=pdb
#
# --- PDB USER NAME ---
#
pdb_username=sa
#
# --- PDB PASSWORD ---
#
pdb_password=
#
# --- RDS NAME ---
#
replicate_ds=RDS
#
# --- RDB NAME ---
#
replicate_db=rdb
#
# --- RDB USER NAME ---
#
rdb_username=sa
#
# --- RDB PASSWORD ---
#
rdb_password=
#
# --- PORT FOR RS ---
#
rs_port=11754
#
# --- PORT FOR RSSD ---
#
rssd_port=11755
#
# --- PORT FOR PDS ---
#
pds_port=20000
#
# --- PORT FOR RDS ---
```

```
#
rds_port=20001
#
#########################################################################
#######
# --- Part 3. transaction profile configuration parameters  --- #
#########################################################################
#######
#
# --- number of transactions to be executed --- #
#
tran_number=100
#
# --- what kind of transction will be executed --- #
#    1."Tran_Profile_1(insert--48% delete--4% update 48%)"
#    2."Tran_Profile_2(insert--30% delete--5% update 65%)"
#    3."Tran_Profile_3(insert--61% delete--2% update 37%)"
#    4."Tran_Profile_LargeTran"
#
tran_option=1
#
#########################################################################
#######
# --- Part 4. system settings --- #
#########################################################################
#######
#
# --- WAIT TIME FOR CONNECTING SERVERS, SPECIFIED BY SECOND(S) ---
#
wait_time=10
```

配置参考环境

在构建参考复制环境后，可以使用 **config** 参数和配置文件执行 **refimp** 脚本以在参考主数据库和复制数据库上创建表和存储过程，并在参考 Replication Server 上创建数据库复制定义和预订。

输入：

```
refimp config -f config_file
```

可以使用 *config_file* 指定配置文件的名称和位置，您可以在该文件中指定一些参数。

必须使用在构建过程中为 **buildenv** 指定的相同配置文件信息。

如果成功执行 **refimp config**，则会看到：

```
Task succeeded: configuring database replication environment
completed.
```

另请参见

- 为参考环境创建的对象 （第 362 页）

在参考环境中运行性能测试

可以使用 **run** 参数执行 **refimp** 脚本，以便使用数据库级复制自动在主数据服务器上插入、更新和删除数据。

输入：

```
refimp run -f config_file
```

可以使用 *config_file* 指定用于 **refimp config** 的相同配置文件。

如果成功执行 **refimp run**，则会看到：

```
Task succeeded: insert data into primary database completed.
```

从参考环境中获取测试结果

可以使用 **analyze** 参数执行 **refimp** 脚本以收集统计信息和性能信息。

输入：

```
refimp analyze -f config_file
```

可以使用 *config_file* 指定用于 **refimp config** 的相同配置文件。

如果成功执行 **refimp analyze**，则会看到：

```
Task succeeded: fetch performance data completed.
```

从 `$refimp_work_dir/report` 中获取 rs_ticket_history 报告以及监控器和计数器报告，其中 *refimp_work_dir* 是在配置文件中指定的位置。

rs_ticket_history 报告

rs_ticket_history 报告使用每个 Replication Server 模块中的票据报告的时间戳说明票据数据通过每个模块所花的时间。

该报告是由 **rs_ticket** 存储过程生成的。请参见《Replication Server 参考手册》的“RSSD 存储过程”中的“**rs_ticket**”。

您可以通过主数据库和复制数据库中的票据报告的时间计算出总复制持续时间。该报告中包含以下列：

- cnt – 票据序列号。
- pdb_t – 在主数据库中执行 **rs_ticket** 存储过程的时间。
- rdb_t – 票据到达复制数据库的时间。
- ticket – 有关票据的信息，其中包括票据通过每个模块的时间。

示例 rs_ticket_history 报告

```
cnt       pdb_t                          rdb_t
---       -------------------            -------------------
  1       Jan 19 2010  2:17AM            Jan 19 2010  2:17AM

ticket
------------------------------------------------------------
V=2;H1=profile1;H2=start;PDB(pdb)=01/19/10 02:17:19.406;
EXEC(40)=01/19/10 02:17:19.423;B(40)=1332;
DIST(26)=01/19/10 02:17:19.669;
DSI(35)=01/19/10 02:17:19.916;
DSI_T=1;DSI_C=3;RRS=SAMPLE_RS_XIEL


cnt       pdb_t                          rdb_t
---       -------------------            -------------------
  2       Jan 19 2010  2:20AM            Jan 19 2010  2:20AM

ticket
------------------------------------------------------------

V=2;H1=profile1;H2=end;PDB(pdb)=01/19/10 02:20:32.206;
EXEC(40)=01/19/10 02:20:32.211;B(40)=5044893;
DIST(26)=01/19/10 02:20:32.249;DSI(35)=01/19/10 02:20:32.524;
DSI_T=5410;DSI_C=18297;RRS=SAMPLE_RS_XIEL
```

监控器和计数器报告

监控器和计数器报告说明 Replication Server 计数器报告的性能数字，这些计数器监控您在报告期间执行的命令。

示例监控器和计数器报告
这是一个较长的报告；仅显示一个计数器。

注意： 输出右侧包含一些注释，用于对此示例进行解释。它们不是输出的一部分。

```
Comment: refimp
Jan 19 2010 02:17:39:606AM                        *Start time stamp*
Jan 19 2010 02:20:22:576AM                        *End time stamp*
9                                               *No of obs intervals*
0                                             *No of min between obs*
16384                                           *SQM bytes per block*
64                                           *SQM blocks per segment*
AOBJ                                              *Module name*
10305                                             *Instance ID*
11                                                *Instance value*
AOBJ dbo.district                                 *Module name*
AOBJ: Insert command                           *Counter external name*
AOBJInsertCommand                               *Counter display name*
65000, , 10305, 11                              *Counter ID, instance
ID,
                                                            instance value*
ENDOFDATA                                       *EOD for counter*
```

```
AOBJ: Update command                              *Counter external name*
AOBJUpdateCommand                                  *Counter display name*
65000, , 10305, 11                                 *Counter ID, instance
ID,
                                                                 instance value*
Jan 19 2010 02:17:39:606AM, 50, 50,  1,  1       *Dump ts, obs, total,
                                                          last, max*
....
ENDOFDATA                                           *EOD for counter*
```

另请参见

- 使用计数器监控性能 （第 251 页）

关闭参考实现服务器

在清除环境后，可以执行 **cleanenv** 脚本以关闭 Replication Server 和数据服务器。

输入：

```
cleanenv -f config_file
```

可以使用 *config_file* 指定用于 **refimp config** 的相同配置文件。

如果成功执行 **cleanenv**，则会看到：

```
Task succeeded: shut down all the servers.
```

清除参考环境

可以使用 **cleanup** 参数执行 **refimp** 脚本以删除测试数据，然后删除复制定义、预订、表和存储过程以准备进行下一次测试。

输入：

```
refimp  cleanup -f config_file
```

可以使用 *config_file* 指定用于 **refimp config** 的相同配置文件。

如果成功执行 **refimp cleanup**，则会看到：

```
Task succeeded: clean up database replication environment completed.
```

为参考环境创建的对象

参考实现工具集在参考复制环境中创建存储过程、复制定义、预订和表对象。

表 38. 为参考实现创建的存储过程

存储过程	位置
sp_load_warehouse_data	主数据库和复制数据库
sp_load_district_data	主数据库和复制数据库
sp_load_customer_data	主数据库和复制数据库
sp_load_history_data	主数据库和复制数据库
sp_load_item_data	主数据库和复制数据库
sp_load_stock_data	主数据库和复制数据库
sp_load_order_orderline_data	主数据库和复制数据库
sp_load_neworder_data	主数据库和复制数据库
sp_load_data_multi_tran	主数据库和复制数据库
sp_gen_neworder_data	主数据库
sp_gen_payment_data	主数据库
sp_gen_delivery_data	主数据库
sp_gen_neworder_data_large_tran	主数据库
sp_gen_payment_data_large_tran	主数据库
sp_gen_delivery_data_large_tran	主数据库
sp_generator_data_1	主数据库
sp_generator_data_2	主数据库
sp_generator_data_3	主数据库
sp_generator_data_4	主数据库

表 39. 为参考实现创建的复制定义和预订

存储过程	预订
复制定义：**pdbrepdefforrdb**	
预订：**rdbsubforpdb**	**pdbrepdefforrdb**

表 40. 为参考实现创建的表

表	位置
WAREHOUSE	主数据库和复制数据库
DISTRICT	主数据库和复制数据库
CUSTOMER	主数据库和复制数据库
HISTORY	主数据库和复制数据库
NEW_ORDER	主数据库和复制数据库
ORDER	主数据库和复制数据库
ORDER_LINE	主数据库和复制数据库
ITEM	主数据库和复制数据库
Stock	主数据库和复制数据库

表模式

为参考实现创建的表的表模式。

表 41. WAREHOUSE

字段名	字段定义	注释
W_ID	2*W 个唯一 ID	W 是仓库号
W_NAME	可变文本，大小为 10	
W_STREET1	可变文本，大小为 20	
W_STREET2	可变文本，大小为 20	
W_CITY	可变文本，大小为 20	
W_STATE	固定文本，大小为 2	
W_ZIP	固定文本，大小为 9	
W_TAX	数值，4 位数	营业税
W_YTD	数值，12 位数	年初至今结余

键：

- 主键：(W_ID)

表 42. DISTRICT

字段名	字段定义	注释
D_ID	20 个唯一 ID	每个仓库填充 10 个
D_W_ID	2*W 个唯一 ID	
D_NAME	可变文本，大小为 10	
D_STREET1	可变文本，大小为 20	
D_STREET2	可变文本，大小为 20	
D_CITY	可变文本，大小为 20	
D_STATE	固定文本，大小为 2	
D_ZIP	固定文本，大小为 9	
D_TAX	数值，4 位数	营业税
D_YTD	数值，12 位数	年初至今结余
D_NEXT_O_ID	10, 000 个唯一 ID	下一个可用订单号的唯一 ID

键：

- 主键（D_W_ID、D_ID）
- 外键 (D_W_ID) 引用 (W_ID)

表 43. CUSTOMER

字段名	字段定义	注释
C_ID	96, 000 个唯一 ID	每个仓库填充 3, 000 个
C_D_ID	20 个唯一 ID	
C_ID	2*W 个唯一 ID	
C_FIRST	可变文本，大小为 16	
C_MIDDLE	固定文本，大小为 2	
C_LAST	可变文本，大小为 16	
C_STREET1	可变文本，大小为 20	
C_STREET2	可变文本，大小为 20	
C_CITY	可变文本，大小为 20	

字段名	字段定义	注释
C_STATE	固定文本，大小为 2	
C_ZIP	固定文本，大小为 9	
C_PHONE	固定文本，大小为 16	
C_SINCE	日期和时间	注册日期
C_CREDIT	固定文本，大小为 2	信用："GC"= 信用良好，"BC"= 信用不良
C_CREDIT_LIM	数值，12 位数	
C_DISCOUNT	数值，4 位数	
C_BALANCE	带符号的数值，12 位数	
C_YTD_PAYMENT	数值，12 位数	
C_PAYMENT_CNT	数值，4 位数	
C_DELIVERY_CNT	数值，4 位数	
C_DATA	可变文本，大小为 500	备注

键：

- 主键（C_W_ID、C_D_ID、C_ID）
- 外键（C_W_ID、C_D_ID）引用（D_W_ID、D_ID）

表 44. HISTORY

字段名	字段定义	注释
H_C_ID	96, 000 个唯一 ID	
H_C_D_ID	20 个唯一 ID	
H_C_W_ID	2*W 个唯一 ID	
H_D_ID	20 个唯一 ID	
H_W_ID	2*W 个唯一 ID	
H_DATE	日期和时间	
H_AMOUNT	数值，6 位数	
H_DATA	可变文本，大小为 24	

键：

- 主键：无
- 外键（H_C_W_ID、H_C_D_ID、H_C_ID）引用（C_W_ID、C_D_ID、C_ID）
- 外键（H_W_ID、H_D_ID）引用（D_W_ID、D_ID）

表 45. NEW_ORDER

字段名	字段定义	注释
N_O_ID	10, 000, 000 个唯一 ID	
N_D_ID	20 个唯一 ID	
NO_W_ID	2*W 个唯一 ID	

键：

- 主键（NO_W_ID、NO_D_ID、NO_O_ID)
- 外键（NO_W_ID、NO_D_ID、NO_O_ID）引用（O_W_ID、O_D_ID、O_ID）

表 46. ORDER

字段名	字段定义	注释
O_ID	10, 000, 000 个唯一 ID	
O_D_ID	20 个唯一 ID	
O_W_ID	2*W 个唯一 ID	
O_C_ID	96, 000 个唯一 ID	
O_ENTRY_D	日期和时间	
O_CARRIER_ID	10 个唯一 ID 或 Null	
O_OL_CNT	从 5 到 15	
O_ALL_LOCAL	数值，1 位数	

键：

- 主键（O_W_ID、O_D_ID、O_ID)
- 外键（O_W_ID、O_D_ID、O_C_ID）引用（C_W_ID、C_D_ID、C_ID）

表 47. ORDER_LINE

字段名	字段定义	注释
OL_O_ID	10, 000, 000 个唯一 ID	
OL_D_ID	20 个唯一 ID	

字段名	字段定义	注释
OL_W_ID	2*W 个唯一 ID	
OL_NUMBER	15 个唯一 ID	
OL_I_ID	200,000 个唯一 ID	
OL_SUPPLY_W_ID	2*W 个唯一 ID	
OL_DELIVERY_D	日期和时间或 Null	
OL_QUANTITY	数值，2 位数	
OL_AMOUNT	数值，6 位数	
OL_DIST_INFO	固定文本，大小为 24	

键：

- 主键（OL_W_ID、OL_D_ID、OL_O_ID、OL_NUMBER）
- 外键（OL_W_ID、OL_D_ID、OL_O_ID）引用（O_W_ID、O_D_ID、D_ID）
- 外键（OL_SUPPLY_W_ID、OL_I_ID）引用（S_W_ID、S_I_ID）

表 48. ITEM

字段名	字段定义	注释
I_ID	200, 000 个唯一 ID	
I_IM_ID	200, 000 个唯一 ID	
I_NAME	可变文本，大小为 50	
I_PRICE	数值，5 位数	
I_DATA	可变文本，大小为 50	

键：

- 主键 (I_ID)

表 49. STOCK

字段名	字段定义	注释
S_I_ID	200, 000 个唯一 ID	
S_W_ID	2*W 个唯一 ID	
S_QUANTITY	数值，4 位数	

字段名	字段定义	注释
S_DIST_01	固定文本，大小为 24	
S_DIST_02	固定文本，大小为 24	
S_DIST_03	固定文本，大小为 24	
S_DIST_04	固定文本，大小为 24	
S_DIST_05	固定文本，大小为 24	
S_DIST_06	固定文本，大小为 24	
S_DIST_07	固定文本，大小为 24	
S_DIST_08	固定文本，大小为 24	
S_DIST_09	固定文本，大小为 24	
S_DIST_10	固定文本，大小为 24	
S_YTD	数值，8 位数	
S_ORDER_CNT	数值，4 位数	
S_REMOTE_CNT	数值，4 位数	
S_DATA	可变文本，大小为 50	

键：

- 主键（S_W_ID、S_I_ID）
- 外键 (S_W_ID) 引用 (W_ID)
- 外键 (S_I_ID) 引用 (I_ID)

词汇表

复制系统中使用的术语的词汇表。

- **活动数据库** – 在热备份应用程序中，复制到备用数据库的数据库。另请参见*热备份应用程序*。
- **Adaptive Server** – Sybase 11.5 版和更高版本的关系数据库服务器。如果您在配置 Replication Server 时选择 RSSD 选项，Adaptive Server 就会维护 RSSD 数据库中的 Replication Server 系统表。
- **应用程序编程接口 (API)** – 一个预定义的接口，用户或程序通过它相互通信。例如，Open Client 和 Open Server 就是 API，它们在客户端/服务器体系结构中进行通信。RCL，即“复制命令语言”，是 Replication Server API。
- **应用函数** – 与函数复制定义相关联的复制函数，Replication Server 会将它从主数据库传送到预订的复制数据库。此函数将参数值传递给在复制数据库中执行的存储过程。存储过程由维护用户在复制数据库中执行。另请参见*复制函数传递*、*请求函数*和*函数复制定义*。
- **项目** – 可以是发布元素的表或存储过程的复制定义扩展。项目可能包含、也可能不包含用于指定复制数据库接收的行的子集的 **where** 子句。
- **异步过程传递** – 一种将与表复制定义关联的存储过程从源数据库复制到目标数据库的方法。
- **异步命令** – 客户端提交的命令，提交此命令后，客户端可以在接收到完成状态前继续进行其它的操作。复制系统中的许多 Replication Server 命令都是异步命令。
- **原子实现** – 一种实现方法，这种方法使用启用了 holdlock 选项的 **select** 操作，以单个基本操作的形式，通过网络将预订数据从主数据库复制到复制数据库。在数据传输完成之前，不允许对主数据进行任何更改。复制数据既可以作为单个事务应用，或以每事务十行的增量方式应用，这样可确保复制数据库事务日志不会填满。原子实现是 **create subscription** 命令的缺省方法。另请参见*非原子实现*、*批量实现*和*不实现*。
- **autocorrection** – 自动更正是一项应用于复制定义的设置，它使用 **set autocorrection** 命令来防止由于在被复制的表的副本中丢失行或出现重复行而发生故障。启用自动更正后，Replication Server 会将每个 update 或 insert 操作转换为一个 delete 操作后跟一个 insert 操作。仅 应为其预订使用非原子实现的复制定义启用自动更正。
- **基类** – 不从父类继承函数字符串的函数字符串类。另请参见*函数字符串类*。
- **位图预订** – 一种基于位图比较对行进行复制的预订。当您创建复制定义时，使用 `int` 数据类型创建列，并将这些列标识为 `rs_address` 数据类型。创建预订时，使用 **where** 子句中的位图比较运算符 (&) 将每个 `rs_address` 列与位掩码进行比较。与预订的位图相匹配的行将进行复制。
- **批量拷入** – 一项功能，可在复制针对 Adaptive Server® Enterprise 12.0 和更高版本的同一表的大批量 **insert** 语句时提高 Replication Server 性能。Replication Server 使用 Open Client™ Open Server™ Bulk-Library 在数据服务器接口 (DSI) 中实现批量拷入功能；DSI 是负责将事务发送到复制数据库的 Replication Server 模块。

批量拷入还改善了预订实现的性能。如果将 **dsi_bulk_copy** 设置为 on 并且每个事务中的 **insert** 命令数超过 **dsi_bulk_threshold**，Replication Server 将使用批量拷入实现预订。

- **批量实现** – 一种实现方法，用于在复制系统的外部初始化复制数据库中的预订数据。例如，可使用介质（如磁带、磁盘、CD-ROM 或光学存储磁盘）从主数据库中传输数据。批量实现涉及从 **define subscription** 开始的一系列命令。对于表复制定义或函数复制定义的预订可以使用批量实现。另请参见*原子实现*、*非原子实现*和*不实现*。
- **集中式数据库系统** – 一种数据库系统，其中的数据由位于中央位置的单个数据库管理系统进行管理。
- **类** – 请参见*错误类*和*函数字符串类*。
- **类树** – 从同一基类派生的一组函数字符串类，由两层或更多层派生类和父类组成。另请参见*函数字符串类*。
- **客户端** – 在客户端/服务器体系结构中连接到服务器的程序。它可以是用户执行的前端应用程序，也可以是作为系统的扩展执行的实用程序。
- **客户端/服务器接口 (C/SI)** – 在客户端/服务器体系结构中执行的程序的 Sybase 接口标准。
- **并发** – 多个客户端共享数据或资源的能力。数据库管理系统中的并发取决于系统是否能保护客户端，以防止因某个客户端正在使用的数据由另一个客户端修改所引发的冲突。
- **连接** – 从 Replication Server 到数据库的连接。另请参见*数据服务器接口 (DSI)* 和*逻辑连接*。
- **连接配置文件** – 在连接配置文件中，您可以使用一组预定义的属性来配置连接。
- **协调的转储** – 通过在复制系统中分发 **rs_dumpdb** 或 **rs_dumptran** 函数，从而在多个节点间保持同步的一组数据库转储或事务转储。
- **数据库** – 为某一特定目的而组织和存放的一组相关的数据表和其它对象。
- **数据库生成号** – 同时存储在数据库以及管理该数据库的 Replication Server 的 RSSD 中，数据库生成号是每个日志记录的原始队列 ID (*qid*) 的第一部分。原始队列 ID 可确保 Replication Server 不处理重复的记录。在执行恢复操作的过程中，您可能需要增大数据库世代号，以便 Replication Server 不会忽略在重新装载数据库后提交的记录。
- **数据库复制定义** – 对一组可创建预订的数据库对象（表、事务、函数、系统存储过程和 DDL）的说明。

  您还可以创建表复制定义和函数复制定义。另请参见*表复制定义*和*函数复制定义*。
- **数据库服务器** – 向客户端提供数据库管理服务的服务器程序（如 Sybase Adaptive Server）。
- **数据定义语言 (DDL)** – 查询语言（如 Transact-SQL）中用于描述数据库中的数据及其相互关系的命令集。Transact-SQL 中的 DDL 命令包括那些使用 **create**、**drop** 和 **alter** 关键字的命令。

- **数据操作语言 (DML)** – 查询语言（如 Transact-SQL）中用于对数据进行操作的命令集。Transact-SQL 中的 DML 命令包括 **select**、**insert**、**update** 和 **delete**。
- **数据服务器** – 一种服务器，其客户端接口符合 Sybase 客户端/服务器接口标准，并提供维护数据库中复制表的物理表示所需的功能。数据服务器通常是数据库服务器，但它们也可以是具有 Replication Server 所需的接口和功能的任何数据存储库。
- **数据服务器接口 (DSI)** – 与 Replication Server 和数据库之间的连接相对应的 Replication Server 线程。DSI 线程将事务从 DSI 外发队列提交到复制数据服务器。它们由一个调度程序线程和一个或多个执行程序线程组成。调度程序线程按照提交顺序将事务分组，然后将它们分派给执行程序线程。执行程序线程将函数映射到函数字符串，然后在复制数据库中执行这些事务。DSI 线程使用 Open Client 方式连接到数据库。另请参见*外发队列*和*连接*。
- **数据源** – 数据库管理系统 (DBMS) 产品（如关系数据服务器或非关系数据服务器）、驻留在 DBMS 中的数据库以及通信方法（用于从复制系统的其它部分访问 DBMS）的特定组合。另请参见*数据库*和*数据服务器*。
- **决策支持应用程序** – 一种数据库客户端应用程序，其特点是能够进行即席查询、报告、计算操作并拥有很少的数据更新事务。
- **声明的数据类型** – 从 Replication Agent 传递到 Replication Server 的值的数据类型：
  - 如果 Replication Agent 向 Replication Server 传递一个基本 Replication Server 数据类型（如 `datetime`），则声明的数据类型为基本数据类型。
  - 否则，声明的数据类型必须为主数据库中原始数据类型的 UDD。
- **缺省函数字符串** – 缺省情况下为以下类提供的函数字符串：系统提供的类 `rs_sqlserver_function_class` 和 `rs_default_function_class`，以及从这些类直接或间接继承函数字符串的类。另请参见*函数字符串*。
- **取消实现** – 可选进程，当删除一个预订时，该进程可将其它预订未使用的特定行从复制数据库中删除。
- **派生类** – 从父类继承函数字符串的函数字符串类。另请参见*函数字符串类*和*父类*。
- **直接路由** – 用于将消息直接从源 Replication Server 发送到目标 Replication Server 的路由，中间无需经过任何 Replication Server。另请参见*间接路由*和*路由*。
- **磁盘分区** – 请参见*分区*。
- **分布式数据库系统** – 一种将数据存储于网络上的多个数据库中的数据库系统。数据库可由同一类型的数据服务器（如 Adaptive Server）或异构数据服务器来管理。
- **分发器** – 一种 Replication Server 线程 (DIST)，用于帮助确定入站队列中每个事务的目标。
- **转储标记** – 执行转储时由 Adaptive Server 写入数据库事务日志的消息。在热备份应用程序中，当您使用活动数据库中的数据初始化备用数据库时，可以指定 Replication Server 使用转储标记，以确定在事务流中从何处开始应用备用数据库中的事务。另请参见*热备份应用程序*。
- **嵌入式 Replication Server 系统数据库 (ERSSD)** – 用于存储 Replication Server 系统表的 SQL Anywhere (SA) 数据库。您可以选择是将 Replication Server 系统表存

储在 ERSSD 上还是存储在 Adaptive Server RSSD 上。另请参见 *Replication Server 系统数据库 (RSSD)*。

- **Enterprise Connect Data Access (ECDA)** – 一组集成的软件应用程序和连接工具，用于访问异构数据库环境中的数据，如各种基于 LAN 的非 ASE 数据源以及大型机数据源。
- **ExpressConnect for Oracle** – 一组可用于在 Replication Server 和 Oracle 数据库之间提供直接通信的库。
- **错误操作** – Replication Server 对数据服务器错误的响应。可能的 Replication Server 错误操作有 **ignore**、**warn**、**retry_log**、**log**、**retry_stop** 和 **stop_replication**。错误操作将指派给特定的数据服务器错误。
- **错误类** – 用于指定数据库的数据服务器错误操作集合的名称。
- **例外日志** – 三个一组的 Replication Server 系统表，其中包含有关数据服务器上的失败事务的信息。日志中的事务必须由用户或智能应用程序进行解析。可以使用 **rs_helpexception** 存储过程查询例外日志。
- **故障切换** – Sybase 故障切换使您能够将两个 12.0 版或更高版本的 Adaptive Server 配置为协同服务器。如果主协同服务器失败，则辅助协同服务器可接管该服务器的设备、数据库和连接。

  有关 Sybase 故障切换在 Adaptive Server 中的工作原理的更多详细信息，请参见《在高可用性系统中使用 Sybase 故障切换》，此文档是 Adaptive Server Enterprise 文档集的一部分。
- **容错** – 系统在其一个或多个组件发生故障时仍能继续正确运行的能力。
- **函数** – 表示数据服务器操作（如 insert、delete、select 或 begin 事务）的 Replication Server 对象。Replication Server 将这些操作作为函数分发到其它 Replication Server。每个函数由函数名和一组数据参数组成。为了在目标数据库中执行函数，Replication Server 使用函数字符串将函数转换为某类数据库的一个命令或一组命令。另请参见*用户定义的函数*和*复制函数传递*。
- **函数复制定义** – 对复制函数传递中使用的复制函数的说明。函数复制定义由 Replication Server 维护，包括有关待复制的参数以及受影响数据的主版本位置的信息。有两类函数复制定义：“应用”和“请求”。另请参见*复制函数传递*。
- **函数作用域** – 函数的影响范围。函数具有复制定义作用域或函数字符串类作用域。具有复制定义作用域的函数是针对特定的复制定义而定义的，因此无法将其应用于其它复制定义。具有函数字符串类作用域的函数只针对函数字符串类定义一次，并且仅在该类中可用。
- **函数字符串** – Replication Server 用于将数据库命令映射到数据服务器 API 的字符串。只有 **rs_select** 和 **rs_select_with_lock** 函数的字符串包含输入模板，用于匹配函数字符串与数据库命令。所有函数的字符串均包含输出模板，用于格式化目标数据服务器的数据库命令。
- **函数字符串类** – 用于指定数据库连接的函数字符串的命名集合。函数字符串类包括那些随 Replication Server 提供的函数字符串类以及那些您创建的函数字符串类。函数字符串类可通过函数字符串继承共享函数字符串定义。系统提供的三个函数字符串类是：`rs_sqlserver_function_class`、

rs_default_function_class 和 rs_db2_function_class。另请参见*基类*、*类树*、*派生类*、*函数字符串继承* 和 *父类*。

- **函数字符串继承** – 在类之间共享函数字符串定义的能力，派生类凭借此能力从父类继承函数字符串。另请参见*派生类*、*函数字符串类* 和 *父类*。
- **函数字符串变量** – 函数字符串中使用的标识符，用于表示将在运行时被替代的值。函数字符串中的变量用问号 (?) 括起。它们表示列值、函数参数、系统定义的变量或用户定义的变量。
- **函数预订** – 对函数复制定义（用于应用函数传递和请求函数传递）的预订。
- **网关** – 连接软件，用于实现两个或多个计算机系统与不同网络体系结构进行通信。
- **生成号** – 请参见*数据库生成号*。
- **异构数据服务器** – 在分布式数据库系统中一起使用的多个供应商的数据服务器。
- **休眠模式** – 一种 Replication Server 状态。在此状态下，系统将拒绝除 **admin** 和 **sysadmin** 命令之外的所有 DDL 命令；挂起所有路由和连接；挂起大多数服务线程（如 DSI 和 RSI）；注销 RSI 和 RepAgent 用户并且不允许他们登录。此模式可在路由升级过程中使用，并且可为 Replication Server 开启以便调试问题。
- **高可用性 (HA)** – 停机时间非常少。提供 HA 的计算机系统通常提供 99.999% 的可用性，或大约每年 5 分钟的意外停机时间。
- **高容量自适应复制 (HVAR)** – 编译一组 **insert**、**delete** 和 **update** 操作，可生成净结果以及后续的将净结果批量应用到复制数据库。
- **热备份应用程序** – 一种数据库应用程序，通过此程序可将备用数据库置于服务状态而无需中断客户端应用程序，并且不会丢失任何事务。另请参见*热备份应用程序*。
- **ID Server** – 复制系统中有一个 Replication Server 是 ID Server。除了执行 Replication Server 的常规任务外，ID Server 还向复制系统中的每个 Replication Server 和数据库指派唯一的 ID 号，并维护复制系统的版本信息。
- **入站队列** – 用于将消息从 Replication Agent 假脱机到 Replication Server 的稳定队列。
- **间接路由** – 用于通过一个或多个中间 Replication Server 将消息从源 Replication Server 发送到目标 Replication Server 的路由。另请参见*直接路由* 和*路由*。
- **interfaces 文件** – 包含为 Sybase 客户端/服务器体系结构中的服务器程序定义网络访问信息的条目的文件。服务器程序可以包括 Adaptive Server、网关、Replication Server 和 Replication Agent。通过 interfaces 文件条目，客户端和服务器便可以在网络上彼此相连。
- **延迟** – 对将首先应用于主数据库的数据修改操作分发到复制数据库所需时间的度量。此时间包括 Replication Agent 处理、Replication Server 处理和网络开销。
- **局域网 (LAN)** – 由计算机和设备（如打印机和终端）组成的系统，这些计算机和设备为共享数据和设备而通过线路连接起来。
- **定位符值** – 存储在 Replication Server 的 RSSD 的 rs_locater 表中的值，用于标识在复制期间 Replication Server 从每个以前的节点接收和确认的最新日志事务记录。

- **逻辑连接** – 一种数据库连接，Replication Server 将它映射到热备份环境中的活动数据库和备用数据库的连接。另请参见*连接*和*热备份应用程序*。
- **登录名** – 用户或系统组件（如 Replication Server）登录到数据服务器、Replication Server 或 Replication Agent 所使用的名称。
- **日志传送语言 (LTL)** – 复制命令语言 (RCL) 的子集。Replication Agent（如 RepAgent）使用 LTL 命令向 Replication Server 提交它从主数据库事务日志中检索到的信息。
- **Log Transfer Manager (LTM)** – 用于 Sybase SQL Server 的 Replication Agent 程序。另请参见 *Replication Agent* 和 *RepAgent 线程*。
- **维护用户** – Replication Server 用于维护复制数据的数据服务器登录名。在大多数应用程序中，不会复制维护用户事务。
- **实现** – 将预订指定的数据从主数据库复制到复制数据库，从而对复制表进行初始化的进程。可通过网络传送复制数据，或者，如果预订涉及大量数据，则最初可从介质装载复制数据。另请参见*原子实现*、*批量实现*、*不实现*和*非原子实现*。
- **实现队列** – 一种稳定队列，用于假脱机与正在实现或取消实现的预订相关的消息。
- **缺失行** – 在表的复制副本中缺失却出现在主表中的行。
- **混合版本系统** – 由不同软件版本的 Replication Server 构成的复制系统。根据其软件版本和节点版本的不同，其功能也不相同。只有版本为 11.0.2 或更高版本的系统才支持混合版本。

  例如，包含 Replication Server 11.5 版或更高版本和 11.0.2 版的复制系统就是混合版本系统。包含版本低于 11.0.2 的 Replication Server 的复制系统不是混合版本系统，因为任何更高版本的 Replication Server 均受到系统版本的限制，无法使用某些新的功能。另请参见*节点版本*和*系统版本*。
- **更多列** – 复制定义中超过 250 但限于 1024 的列。Replication Server 12.5 版和更高版本支持更多列。
- **多节点可用性 (MSA)** – 将数据库对象（表、函数、事务、系统存储过程和 DDL）从主数据库复制到复制数据库的方法。另请参见*数据库复制定义*。
- **Multi-Path Replication™** – Replication Server 功能，通过启用从源数据库到目标数据库的并行数据路径来提高性能。可以在热备份和多节点可用性 (MSA) 环境中配置多路径复制。这些路径独立处理数据，并适合并行处理没有事务一致性要求的数据集。而仍在路径内维护数据一致性，但不跨不同路径遵循提交顺序。
- **名称空间** – 对象名在其中必须唯一的范围。
- **非原子实现** – 以单个操作的形式（不使用锁定），通过网络将预订数据从主数据库复制到复制数据库的实现方法。在数据传输过程中允许对主表进行更改，这可能导致复制数据库和主数据库之间暂时不一致。数据按每个事务十行的增量方式应用，这样可以确保不会填满复制数据库事务日志。非原子实现是 **create subscription** 命令的可选方法。另请参见*自动更正*、*原子实现*、*不实现*和*批量实现*。
- **基于网络的安全性** – 跨网络的安全数据传输。Replication Server 支持提供用户身份验证、统一登录和 Replication Server 间安全消息传输的第三方安全性机制。

- **不实现** – 一种可使您在复制节点中已存在预订数据时创建预订的实现方法。使用带有 **without materialization** 子句的 **create subscription** 命令。可以使用此方法创建对表复制定义和函数复制定义的预订。另请参见*原子实现*和*批量实现*。
- **联机事务处理 (OLTP) 应用程序** – 一种数据库客户端应用程序，具有频繁处理涉及数据修改（插入、删除和更新）的事务的特点。
- **原始队列 ID (qid)** – `qid` 由 RepAgent 构成，它唯一地标识传递到 Replication Server 的每个日志记录。它包括 `date`、`timestamp` 和数据库生成号。另请参见*数据库生成号*。
- **孤行** – 表的复制副本中与活动预订不匹配的行。
- **外发队列** – 用于假脱机消息的稳定队列。DSI 外发队列将消息假脱机到复制数据库。RSI 外发队列将消息假脱机到复制 Replication Server。
- **并行 DSI** – 对数据库连接进行配置，以便使用以并行方式运行的多个 DSI 线程（而不是单个 DSI 线程）将事务应用到复制数据服务器。另请参见*连接*和*数据服务器接口 (DSI)*。
- **参数** – 对过程执行时提供的值进行表示的标识符。参数名在函数字符串中以 @ 字符作为前缀。当从函数字符串中调用过程时，Replication Server 将参数值原样传递给数据服务器。另请参见*可搜索参数*。
- **父类** – 派生类从中继承函数字符串的函数字符串类。另请参见*函数字符串类*和*派生类*。
- **分区** – Replication Server 用于稳定队列存储的原始磁盘分区或操作系统文件。操作系统文件仅在测试环境中使用。
- **物理连接** – 请参见*连接*。
- **主数据** – 复制系统中一组数据的确定版本。主数据保存在一台对所有预订该数据的 Replication Server 都是已知的数据服务器上。
- **主数据库** – 包含要通过复制系统复制到其它数据库的数据的任何数据库。
- **主段** – 表的水平段，其中含有一个行集的主版本。
- **主键** – 唯一地标识各个行的一组表列。
- **主节点** – 定义函数字符串类或错误类的 Replication Server。请参见*错误类*和*函数字符串类*。
- **主体用户** – 启动应用程序的用户。使用基于网络的安全性时，Replication Server 作为主体用户登录到远程服务器。
- **配置文件** – 在配置文件中，您可以使用一组预定义的属性来配置连接。
- **投影** – 表的垂直片，表示表列的子集。
- **发布** – 同一主数据库中的一组项目。使用发布可以收集相关的表和/或存储过程的复制定义，然后将它们作为一个组进行预订。在源 Replication Server 将复制定义作为项目收集在发布中，然后在目标 Replication Server 通过发布预订来预订它们。另请参见*项目*和*发布预订*。
- **发布预订** – 对发布的预订。另请参见*项目*和*发布*。
- **已发布的数据类型** – 在复制数据服务器中完成列级转换之后（如果有类级别转换，则在类级别转换之前）列的数据类型。发布的数据类型必须是 Replication Server

基本数据类型，或者是目标数据服务器中数据类型的 UDD。如果复制定义中省略了发布的数据类型，则其缺省值为声明的数据类型。

- **查询** – 在数据库管理系统中，查询是检索符合一组给定标准的数据的请求。SQL 数据库语言包括用于查询的 **select** 命令。
- **停顿状态** – 停顿的复制系统是指所有更新已被传播到其目标的复制系统。某些 Replication Server 命令或过程要求首先停顿复制系统。
- **带引号的标识符** – 对象名称，这些名称包含特殊字符（如空格和非字母数字字符）、以字母以外的字符开头或者与保留字相对应，需要用双引号字符将其引起来才能正确进行分析。
- **实时装载 (RTL)** – 高容量自适应复制 (HVAR) 到 Sybase IQ 数据库。使用相关的命令和过程可将 HVAR 更改应用到 Sybase IQ 复制数据库。请参见*高容量自适应复制*。
- **远程过程调用 (RPC)** – 执行驻留在远程服务器中的过程的请求。执行过程的服务器可以是 Adaptive Server、Replication Server 或使用 Open Server 创建的服务器。请求可以源自以上任何服务器，也可源自客户端应用程序。RPC 请求格式是 Sybase Client/Server Interfaces 的一部分。
- **RepAgent 线程** – Adaptive Server 数据库的 Replication Agent。RepAgent 是一个 Adaptive Server 线程；它将事务日志信息从主数据库传送到 Replication Server，以便分配到其它数据库。
- **复制数据库** – 任何包含通过复制系统从其它数据库复制的数据的数据库。
- **复制函数传递** – 一种复制方法，这种方法可将与函数复制定义相关联的存储过程从源数据库复制到目标数据库。另请参见*应用函数*、*请求函数* 和*函数复制定义*。
- **复制存储过程** – 使用 **sp_setrepproc** 或 **sp_setreplicate** 系统过程标记为复制的 Adaptive Server 存储过程。复制存储过程可以与函数复制定义或表复制定义关联。另请参见*复制函数传递* 和*异步过程传递*。
- **复制表** – 由 Replication Server 在多个位置的数据库中进行部分或整体维护的表。该表具有一个主版本，它使用 **sp_setreptable** 或 **sp_setreplicate** 系统过程标记为复制；所有其它版本都是复制的副本。
- **Replication Agent** – 一个程序或模块，它将代表对主数据所作修改的事务日志信息从数据库服务器传送到 Replication Server，以分配到其它数据库。RepAgent 是 Adaptive Server 数据库的 Replication Agent。
- **复制命令语言 (RCL)** – 用于管理 Replication Server 中的信息的命令。
- **复制定义** – 通常是对可创建预订的表的说明。复制定义由 Replication Server 维护，包括有关要复制的列以及表的主版本位置的信息。

  您也可以创建函数复制定义；有时，术语“表复制定义”被用来区分表复制定义和函数复制定义。另请参见*函数复制定义*。
- **Replication Server** – Sybase 服务器程序，它通常维护 LAN 上的复制数据，并处理从相同 LAN 或 WAN 上的其它 Replication Server 接收的数据事务。
- **Replication Server 接口 (RSI)** – 登录到目标 Replication Server 并将命令从 RSI 出站稳定队列传送到目标 Replication Server 的线程。每个目标 Replication Server 均

有一个 RSI 线程，用于从主 Replication Server 或中间 Replication Server 接收命令。另请参见*外发队列*和*路由*。

- **Replication Monitoring Services (RMS)** – 使用 Sybase Unified Agent Framework (UAF) 构建的一个小型 Java 应用程序，用于监控应用程序环境并进行故障排除。
- **复制系统管理员** – 在 Replication Server 中管理例行操作的系统管理员。
- **Replication Server 系统数据库 (RSSD)** – 含有 Replication Server 系统表的 Adaptive Server 数据库。您可以选择是将 Replication Server 系统表存储在 RSSD 上，还是存储在 SQL Anywhere (SA) ERSSD 上。另请参见*嵌入式 Replication Server 系统数据库 (ERSSD)*。
- **Replication Server 系统 Adaptive Server** – 具有包含 Replication Server 系统表 (RSSD) 的数据库的 Adaptive Server。
- **复制系统** – 一种数据处理系统，数据可通过此系统在多个数据库中进行复制，以方便远程用户，使其可以在本地访问数据。尤其是指基于 Replication Server 并包含 Replication Agent 和数据服务器等其它组件的复制系统。
- **复制系统域** – 使用相同 ID Server 的所有复制系统组件。
- **请求函数** – 与函数复制定义关联的复制函数，Replication Server 将它从主数据库传递到复制数据库。此函数将参数值传递给在复制数据库中执行的存储过程。由在主节点执行存储过程的相同用户在复制节点执行存储过程。另请参见*复制函数传递*、*请求函数*和*函数复制定义*。
- **resync marker** – 当您在 resync 模式中重新启动 Replication Agent 时，Replication Agent 会向 Replication Server 发送 resync database 标记来指示正在进行重新同步工作。resync 标记是 Replication Agent 在发送任何 SQL 数据定义语言 (DDL) 或数据操作语言 (DML) 事务之前发送的第一个消息。
- **路由** – 从源 Replication Server 到目标 Replication Server 的单向消息流。路由在 Replication Server 之间传送数据修改命令（包括用于 RSSD 的命令）和复制函数或存储过程。另请参见*直接路由*和*间接路由*。
- **路由版本** – 路由的源 Replication Server 和目标 Replication Server 的节点版本号中较低的版本号。Replication Server 11.5 版和更高版本使用路由版本号来确定发送到复制节点的数据。另请参见*节点版本*。
- **行迁移** – 一种进程，此进程基于与预订的 **where** 子句中的值的比较结果，在表的主版本中行的列值更改时在表的复制版本中插入或删除相应的行。
- **SQL Server** – Sybase 关系数据库 11.5 版之前的服务器。
- **SQL 语句复制** – 在 SQL 语句复制过程中，Replication Server 接收修改了主数据的 SQL 语句，而不是事务日志中的各个行更改。Replication Server 将 SQL 语句应用于复制节点。RepAgent 同时发送 SQL 数据操纵语言 (DML) 和各个行更改。根据您的配置，Replication Server 选择单行更改日志复制或 SQL 语句复制。
- **模式** – 数据库的结构。DDL 命令和系统过程更改存储在数据库中的系统表。如果您使用的是 Replication Server 11.5 版或更高版本和 Adaptive Server 11.5 版或更高版本，则可以将受支持的 DDL 命令和系统过程复制到备用数据库。
- **可搜索列** – 可在预订或项目的 **where** 子句中指定的复制表中的列，用于限制在节点上复制的行。

- **可搜索参数** – 可在预订的 **where** 子句中指定的复制存储过程中的参数，用来帮助确定是否应复制该存储过程。另请参见*参数*。
- **secondary truncation point** – 请参见*截断点*。
- **节点** – 至少包含 Replication Server、数据服务器和数据库（还可能包含 Replication Agent）的安装，通常位于分散的地理位置。每个节点上的组件通过 WAN 连接到复制系统中其它节点上的组件。另请参见*主节点*。
- **节点版本** – 单个 Replication Server 的版本号。将节点版本设置为特定的级别后，Replication Server 将启用该级别所特有的功能，并且不允许降级。另请参见*软件版本*、*路由版本* 和 *系统版本*。
- **软件版本** – 单个 Replication Server 的软件版本的版本号。另请参见 *节点版本* 和 *系统版本*。
- **稳定队列管理器 (SQM)** – 管理稳定队列的线程。无论是入站队列还是外发队列，Replication Server 访问的每个稳定队列都有一个稳定队列管理器 (SQM) 线程。
- **稳定队列事务 (SQT) 接口** – 按提交顺序重新组合事务命令的线程。稳定队列事务 (SQT) 接口线程从入站稳定队列中读取事务，按提交顺序放置事务，然后将事务发送到分发器 (DIST) 线程或 DSI 线程，具体取决于哪个线程要求对事务进行 SQT 排序。
- **稳定队列** – Replication Server 用来存储发往路由或数据库连接的信息的存储转发队列。写入稳定队列的消息在可以传递到目标 Replication Server 或数据库之前，一直保留在存储转发队列中。Replication Server 使用它的磁盘分区生成稳定队列。另请参见*入站队列*、*外发队列* 和 *实现队列*。
- **独立模式** – 一种特殊的 Replication Server 模式，用于启动恢复操作。
- **备用数据库** – 在热备份应用程序中，从活动数据库接收数据修改并充当该数据库的备份的数据库。另请参见*热备份应用程序*。
- **存储过程** – 按某一名称存储在 Adaptive Server 数据库中的 SQL 语句和可选控制流语句的集合。与 Adaptive Server 一起提供的存储过程称为系统过程。有些用于查询 RSSD 的存储过程包含在 Replication Server 软件中。
- **预订** – 请求 Replication Server 在指定的位置维护复制数据库中的表（或表中的行集）的复制副本。还可以预订函数复制定义，以用于复制存储过程。
- **预订取消实现** – 请参见*取消实现*。
- **预订实现** – 请参见*实现*。
- **预订迁移** – 请参见*行迁移*。
- **Sybase Central** – 一种图形工具，提供用于管理 Sybase 和 Powersoft 产品的公用界面。Replication Server 将 Replication Manager 作为 Sybase Central 插件。另请参见 *Replication Monitoring Services (RMS)*。
- **对称多重处理 (SMP)** – 在多处理器平台上，应用程序的线程并行运行的能力。Replication Server 支持 SMP，这可以提高服务器的性能和效率。
- **同步命令** – 客户端只有在收到完成状态后才认为完成的命令。
- **系统函数** – 一种预定义函数，属于 Replication Server 产品的一部分。不同的系统函数协调复制活动（如 **rs_begin**）或者执行数据处理操作（如 **rs_insert**、**rs_delete** 和 **rs_update**）。

- **系统提供的类** – Replication Server 提供错误类 `rs_sqlserver_error_class` 以及函数字符串类 `rs_sqlserver_function_class`、`rs_default_function_class` 和 `rs_db2_function_class`。对于系统提供的函数字符串类以及任何从这些类直接或间接继承的派生类，其函数字符串是自动生成的。另请参见*错误类*和*函数字符串类*。
- **系统版本** – 复制系统的版本号，它表示启用了新功能的版本（对于 Replication Server 11.0.2 版或更早的版本），低于该版本的 Replication Server 不能降级或安装。对于 Replication Server 11.5 版，如果要使用某些新功能，节点版本必须为 1150，系统版本必须至少为 1102。另请参见*混合版本系统*、*节点版本*和*软件版本*。
- **表复制定义** – 请参见*复制定义*。
- **表预订** – 对表复制定义的预订。
- **线程** – 在 Replication Server 中运行的进程。建立在 Sybase Open Server 之上，Replication Server 具有多线程体系结构。每个线程执行特定的功能，例如管理用户会话，接收来自 Replication Agent 或其它 Replication Server 的消息，或者将消息应用到数据库。另请参见*数据服务器接口 (DSI)*、*分配器*和 *Replication Server 接口 (RSI)*。
- **事务** – 对语句进行分组，以便将它们按一个单元处理的机制：或者执行组中的所有语句，或者不执行组中的所有语句。
- **Transact-SQL** – 用于 Adaptive Server 的关系数据库语言。它基于标准 SQL（结构化查询语言），具有 Sybase 扩展。
- **截断点** – 包含主数据的 Adaptive Server 数据库有一个活动截断点，用以标记 Adaptive Server 完成处理的事务日志位置。这就是主截断点。

  Adaptive Server 数据库的 RepAgent 维护辅助截断点，该截断点标记将日志中已成功提交到 Replication Server 的部分与尚未提交的部分分开的事务日志位置。辅助截断点可确保每个操作在它的日志部分被截断之前进入复制系统。
- **用户定义的函数** – 允许创建自定义应用程序的函数，这些应用程序使用 Replication Server 在复制系统的节点之间分发复制函数或异步存储过程。在复制函数传递中，当您创建函数复制定义时，Replication Server 会自动创建用户定义的函数。
- **变量** – 请参见*函数字符串变量*。
- **版本** – *混合版本系统*

  请参见*混合版本系统*、*节点版本*、*软件版本*和*系统版本*。
- **热备份应用程序** – 利用 Replication Server 为数据库（称为活动数据库）维护备用数据库的应用程序。如果活动数据库失败，Replication Server 和客户端应用程序可以切换到备用数据库。
- **广域网 (WAN)** – 与数据通信线路连接在一起的局域网 (LAN) 系统。
- **宽列** – 复制定义中包含宽度大于 255 字节的 `char`、`varchar`、`binary`、`varbinary`、`unichar`、`univarchar` 或 Java `inrow` 数据的列。Replication Server 12.5 版和更高版本支持宽列。

- **宽数据** – 宽数据行，可达到数据服务器中数据页的宽度。Adaptive Server 支持的页大小为 2K、4K、8K 和 16K。Replication Server 12.5 版和更高版本支持宽数据。
- **宽消息** – 大于 16K 并且跨块的消息。Replication Server 12.5 版和更高版本支持宽消息。

获取帮助及其它信息

使用 Sybase 入门 CD、产品文档站点和联机帮助来了解关于此产品版本的更多信息。

- Getting Started CD（或下载） – 包含 PDF 格式的发行公告和安装指南，也可能包含其它文档或更新信息。
- 位于 http://sybooks.sybase.com/ 上的产品文档 – 是 Sybase 文档的在线版本，您可以使用标准 Web 浏览器进行访问。您可以在线浏览文档，也可以采用 PDF 格式进行下载。除产品手册外，该网站还包含指向 EBF/维护、 技术文档、案例管理、已解决的案例、社区论坛/新闻组和其它资源的链接。
- 产品中的联机帮助（如果有）。

要阅读或打印 PDF 文档，您需要 Adobe Acrobat Reader，可以从 Adobe Web 站点免费下载。

注意： 产品文档网站可能会提供更新的发行公告，其中包含在产品发布后增加的重要产品或文档信息。

技术支持部门

获得 Sybase 产品支持。

如果贵组织为此产品购买了支持合同，则您的一个或多个同事将被指定为授权支持联系人。如果您有任何问题，或者在安装过程中需要帮助，请指定专人联系您所在地区的 Sybase 技术支持部门或 Sybase 子公司。

下载 Sybase EBF 和维护报告

可以从 Sybase 网站获得 EBF 和维护报告。

1. 将 Web 浏览器定位到 http://www.sybase.com/support。
2. 从菜单栏或滑出菜单中的**“支持”**下，选择**“EBF/维护”**。
3. 如果出现提示，请输入您的 MySybase 用户名和密码。
4. （可选）从**“显示”**下拉列表中选择过滤器，然后选择时间范围并单击**“开始”**。
5. 选择产品。

   挂锁图标表示您不具有特定 EBF/维护版本的下载权限，因为您未注册成为授权支持联系人。如果您尚未注册，但拥有您的 Sybase 代表提供的或通过您的支持联系人提供的有效信息，请单击**“我的帐户”**向您的 MySybase 配置文件添加“技术支持联系人”。

6. 单击“信息”图标以显示 EBF/维护报告，或者单击产品说明以下载该软件。

Sybase 产品和组件认证

认证报告检验 Sybase 产品在特定平台上的性能。

查找有关认证的最新信息：

- 有关合作伙伴产品认证，请转至 http://www.sybase.com/detail_list?id=9784
- 有关平台认证，请转至 http://certification.sybase.com/ucr/search.do

创建 MySybase 配置文件

MySybase 是一项免费服务，它允许您创建 Sybase 网页的个人化视图。

1. 转至 http://www.sybase.com/mysybase。
2. 单击“立即注册”。

辅助功能特性

辅助功能可确保所有用户（包括残障人士）都能访问电子信息。

Sybase 产品文档采用设计为实现辅助功能的 HTML 版本。

视力受损的用户可以使用自适应技术（如屏幕阅读器）浏览在线文档，或者使用屏幕放大器查看文档。

Sybase HTML 文档已经过测试，符合《美国康复法》第 508 条的辅助功能要求。符合第 508 条的文档一般也符合非美国地区的辅助功能指导原则，如针对网站的 World Wide Web 协会 (W3C) 原则。

注意： 为优化使用性能，您可能需要对辅助工具进行配置。某些屏幕阅读器按照大小写来辨别文本，例如将“ALL UPPERCASE TEXT”看作首字母缩写，而将“MixedCase Text”看作单词。您可能会发现按语法约定来配置工具更为方便。有关工具的信息，请查阅相关文档。

有关 Sybase 如何支持辅助功能的信息，请参见“Sybase 辅助功能”网站： http://www.sybase.com/products/accessibility。该网站包括有关第 508 条和 W3C 标准的信息的链接。

您可以在产品文档中找到更多有关辅助功能特性的信息。

索引

符号

“稳定队列管理器”(SQM) 线程 106

A

abort switch 命令 80
activate subscription 命令
    带 suspension at replicate only 子句 101
    带 suspension 子句 101
Adaptive Server
    错误处理 272
    完全增量编译, 启用 206
    重新同步复制数据库 321
Adaptive Server 监控表
    用于 SQL 语句复制 193
    用于多个复制路径 245
admin config 命令 172
admin logical_status 命令 83
admin show connection, 'primary' 配置参数 227
admin show connection, 'replicate' 配置参数 224
admin sqm_readers 命令 83
admin who 命令
    用于专用路由 245
admin who, dsi 命令 83
admin who, sqm 命令 83
admin 命令 79
    说明 7
allow connections 命令 316
alter connection 命令 171
    将数据库指派给函数字符串类 26
alter function string 命令 36
    替换缺省的函数字符串 333
    映射用户定义的函数 342
alter function 命令 340
alter logical connection 命令 87
alter subscription 配置参数 228
ascii_pack_ibq 122
assign action 命令 272
async_parser 122
安装 Replication Server
    获得 HA 347
    作为数据服务 348
按连接分配
    说明 221
    限制 222

B

batch 配置参数 122
bcp 实用程序 67, 101
block_size to ’value’ with shutdown 配置参数 111
版本守护程序 (dVERSION) 110
版本支持
    重新同步 Adaptive Server 321
绑定对象
    DDL 语句复制 237
    SQL 语句复制 237
    到复制路径 235
    多路径复制 237
    数据库重新同步标记 237
保存间隔
    strict 设置 94, 100
    设置逻辑连接的 93
    说明 286
    为连接设置 289
    为路由设置 288
报警守护程序 (dAlarm) 109
备用数据库 47
    监控添加状态 82
    切换到 76
    添加 66
变量
    函数字符串 31
    系统定义的 32
    修饰符 32
表元数据
    高速缓存 133
表元数据减少
    启用 133
并行 DSI
    OQID 提交堆栈 164
    不频繁的更新冲突 169
    分区规则 159, 168
    分组逻辑 163
    隔离级别 154
    更新冲突 169
    函数字符串 165, 166
    减少争用 168

解决冲突 163
设置参数 149
说明 148
死锁 166
为非 Sybase 复制数据服务器设置隔离级别 155, 169
优点和风险 149
组件 153
最佳性能 167

C

check subscription 命令
执行 switch active 命令后 101
cleanenv 361
Cluster
Sun 345
cmd_direct_replicate 配置参数 122
configure connection 命令, 设置保存间隔 289
configure logical connection 命令 94
设置 DSI 队列保存间隔 94
设置实现队列保存间隔 94
configure replication server 命令 172
create altenate connection 配置参数 223
create connection 命令 26
create error class 268
create function string class 命令 23–25
create function string 命令 33
create function 命令 339
create logical connection 命令 64
create route 命令 244
create subscription 配置参数 227
参考实现 353
cleanenv 361
refimp analyze 359
refimp config 358
refimp run 359
rs_ticket history 报告 359
创建的对象 362
构建环境 354
关闭服务器 361
获取测试结果 359
监控器和计数器报告 360
开始前 354
配置 358
配置文件 355
平台支持 353
所需的组件 353
运行性能测试 359
参数
disk_affinity 144
dsi_cmd_batch_size 143
exec_cmds_per_timeslice 143
exec_sqm_write_request_limit 143
参数, 存储过程
添加到用户定义的函数 340
测试
Replication Server 连接 5
Replication Server 组件 4
查询
用于例外日志系统表 279
查找当前保存间隔 286
抽象计划, 复制 51
出站队列直接复制 139
触发器
在备用数据库中配置 88
创建
函数字符串 33
函数字符串类 23
基本函数字符串类 25
派生的函数字符串类 24
用户定义的函数 339
创建示例
多个复制连接 225
磁盘分区 248
存储过程
rs_delexception 280
rs_helpclass 43
rs_helperror 273
rs_helpexception 278
rs_helpfstring 43
rs_init_erroractions 270
rs_mk_rsids_consistent 304
删除 341
使用 sp_setreplicate 标记为要复制 339
错误
Replication Server 的日志文件 3
标准错误输出 4
错误处理
Replication Server 264
常规 263
数据服务器 267, 273
系统事务 281
指定操作 272
错误类
rs_sqlserver_error_class 268
初始化 270
创建 268

更改主 Replication Server 271
删除 270
指定主节点 269
错误日志文件
Replication Server 3, 264
开始新的 Replication Server 日志文件 266
说明 263
显示当前日志文件名称 266
错误消息
Replication Server 登录名 5
格式 265
系统事务 281
严重级 265

D

db_packet_size 配置参数 112, 123
DB2 数据库, 函数字符串类 11
dbcc settrunc Transact-SQL 命令 294
DDL 语句复制
多路径复制, 绑定对象 237
deferred_name_resolution 配置参数 86
deferred_queue_size 配置参数 112
disk_affinity 配置参数 112, 123, 144
disk_direct_cache_read 配置参数 112
dist_cmd_direct_replicate 123
dist_sqt_max_cache_size 配置参数 123
drop connection 命令 81
drop connection 配置参数 224
drop error class 270
drop function string class 命令 27
drop function string 命令 37
drop function 命令 341
drop logical connection 命令 89
drop route 命令 244
DSI
DSI 线程
用于备用数据库 77
并行 148
处理丢失 315
调度程序 108, 153
挂起以装载批量实现数据 101
检测重复事务 281
说明 105, 108
正在检测丢失 315
执行程序 109, 154
DSI 效率 210
dsi_bulk_copy 连接参数 112, 171, 172
检查值 172
设置值 171
另请参见 批量拷入支持
dsi_bulk_threshold 连接参数 112, 171, 172
检查值 172
设置值 171
另请参见 批量拷入支持
dsi_cdb_max_size 配置参数 124
dsi_cmd_batch_size 参数 143
dsi_cmd_batch_size 配置参数 112, 124
dsi_cmd_prefetch 配置参数 113, 124
dsi_commit_check_locks_intrvl 配置参数 124, 150
dsi_commit_check_locks_log 配置参数 150
dsi_commit_check_locks_max 配置参数 124, 150
dsi_commit_control 配置参数 125, 150
dsi_compile_retry_threshold 配置参数 204
dsi_ignore_underscore_name 配置参数 150
dsi_isolation_level 配置参数 125, 151
dsi_large_xact_size 配置参数 125, 151
dsi_max_cmds_in_batch 151
dsi_max_cmds_in_batch 配置参数 125
dsi_max_xacts_in_group 151
dsi_max_xacts_in_group 配置参数 125
dsi_non_blocking_commit 配置参数 113
dsi_num_large_xact_thread 配置参数 151
dsi_num_large_xact_threads 配置参数 125
dsi_num_threads 配置参数 125, 151
dsi_partitioning_rule 配置参数 126, 151
dsi_row_count_validation 配置参数 274
dsi_serialization_method 配置参数 126, 152
dsi_sqt_max_cache_size 配置参数 127
dsi_text_max_xacts_in_group 配置参数 113
dsi_xact_group_size 配置参数 113, 127
dump database 323
dump database 标记, 发送 323
dump database 命令 72, 290
dump transaction 命令 72, 290
dynamic_sql 配置参数 113
dynamic_sql_cache_management 配置参数 114
dynamic_sql_cache_size 配置参数 113
大事务 154
带引号的标识符
热备份 62
调试
高可用性 351
丢失检测
DSI 丢失 313, 315
SQM 丢失 313

处理丢失 315
防止稳定队列中的假丢失 314
用于热备份应用程序 103
设置日志恢复后 317
正在检测消息 314
重建稳定队列 313
动态 SQL 194
replicate minimal columns, 一使用 195
表级控制 195
配置参数 194
限制 196
独立模式
Replication Server 292, 311, 313
队列 ID 318
队列段, 分配 248
队列块大小
更改 213
建议 213
示例, 简单复制系统 214
示例, 具有中间路由 216
限制 213
队列块大小, 设置 111
队列块大小, 增加 212
多部分复制
并行化 220
多处理器
监控 247
启用 247
多处理器平台 246
多个 RepAgent 连接 229
多个 RepAgent 路径 229
设置内存 230
多个复制定义
和函数字符串 19
多个复制连接
示例, 创建 225
多个复制路径 217
Adaptive Server 监控表 245
monRepLogActivity 监控表 245
monRepScanners 监控表 245
monRepScannersTotalTime 监控表 245
monRepSenders 监控表 245
绑定对象 235
到复制数据库 223
多个 RepAgent 连接 229
多个 RepAgent 路径, 启用 229
多线程 RepAgent 229
解除绑定对象 236
来自主数据库的 226
列出绑定 238
列出复制对象 238
逻辑路径, 删除 234
逻辑路径, 删除元素 234
配置参数 237
切换活动 242
热备份 242
热备份, 切换活动 242
物理路径, 删除 233
专用路由 243
多节点可用性
多路径复制 218, 241
多路径复制 223, 226, 243
DDL 语句复制, 绑定对象 237
MSA 218, 241
number of send buffers 配置参数 231
SQL 语句复制, 绑定对象 237
按对象绑定分配, 说明 220
按连接分配, 说明 221
按连接分配, 限制 222
多个 RepAgent 路径, 设置内存 230
多个连接的复制定义 227
多个连接的预订 227
多线程 RepAgent 231
分配模式 220
逻辑路径, 添加 233
热备份环境 241
设置分布模式 235
数据库重新同步标记, 绑定对象 237
替代连接 241
替代连接, 概念 222
替代逻辑连接 241
物理路径, 添加 232
多线程 RepAgent 229
多线程 RepAgent, 启用 231

E

exec_cmds_per_timeslice 配置参数 114, 127, 143
exec_max_cache_size 配置参数 128
exec_nrm_request_limit 配置参数 114, 128
exec_prs_num_threads 128
exec_sqm_write_request_limit 参数 143
exec_sqm_write_request_limit 配置参数 114, 128

F

方案, 数据库重新同步 324

方案, 数据库重新同步, 热备份 329
方案, 数据库重新同步, 无 resync database 标记支持 327
非 ASE 错误类支持
  本机错误代码 268
  缺省的非 ASE 错误类 268
非原子实现
  在热备份应用程序中 100
分配队列段 248
分配器线程 (DIST)
  禁用 87
  说明 107
分配器线程读取线程效率 212
分区 248
  从丢失或故障中恢复 290, 294
  监控百分比 9
  空间要求 288
分区故障
  恢复 290, 294
分区关系
  alter connection 命令 248
  alter route 命令 248
  rs_diskaffinity 系统表 249
  分配提示 249
  缺省分配 248
分区规则 159, 168
  none 160
  事务名称 161
  用户名 160
  原始开始时间和提交时间 160
服务器
  验证操作 5
服务器用户的 ID
  用于热备份数据库 71
父函数字符串类 23
复制
  数据, 大批量 171
  在备用数据库中配置 88
  主数据库 74
复制 Replication Server
  中的处理 110
复制存储过程
  为复制启用 339
复制定义
  将列发送到备用数据库 98
  热备份 95
复制定义, 为 SQL 语句复制配置 184
复制连接
  替代, 创建 223
  替代, 更改 224
  替代, 显示 224
  替代, 移动预订 228
复制路径
  绑定对象 235
  解除绑定对象 236
  列出绑定 238
  列出复制对象 238
复制数据库
  防止数据丢失 286
复制系统
  错误日志文件 263
  防止数据丢失 286

G

grant 命令 74
高级服务选件 196
高可用性
  安装 Replication Server 以获得 347
  技术概述 346
  脚本 346
  配置 Replication Server 以获得 347
  配置 Sun Cluster 以获得 347
  术语 345
高容量自适应复制 197
高速缓存
  SQL 命令高速缓存中的 LTL 命令 137
  SQM 命令 141
  表元数据 133
  动态高速缓存命令 133
  稳定队列 134
隔离级别 154
隔离级别, 为非 Sybase 数据服务器设置 155
更改
  函数字符串 14
更改复制定义 132
更新函数字符串 36
故障
  数据服务器 263
  网络 263
故障切换, 在 Replication Server 中支持 284

H

ha_failover 配置参数 285
hareg 命令 350
HVAR 197
  admin config 命令 209

dsi_bulk_threshold 202
dsi_cdb_max_size 202
dsi_command_convert 203
dsi_compile_enable 201
dsi_compile_max_cmds 202
dsi_compile_retry_threshold 203
rs_helprep 存储过程 209
编译和批量应用 198
不可编译的命令, 表 200
参照约束 200, 207
处理和限制 199
混合版本支持 210
配置参数 202
平台支持 197
启用 201
完全增量编译 205
系统表支持 210
显示 209
显示表级配置参数 209
显示表引用 209
显示净更改数据库 199
显示数据库级配置参数 209
向后兼容性 210
HVAR 中的 dsi_bulk_threshold 202
HVAR 中的 dsi_cdb_max_size 202
HVAR 中的 dsi_command_convert 203
HVAR 中的 dsi_compile_enable 201
HVAR 中的 dsi_compile_max_cmds 202
HVAR 中的 dsi_compile_retry_threshold 203
HVAR 中的编译和批量应用 198
HVAR 中的参照约束 207
HVAR, 增强的重试机制 204
函数
说明 12
函数复制定义
将参数发送到备用数据库 98
函数字符串
none 45
writetext 45
变量 31
变量, 格式设置 33
变量, 修饰符 32
创建 33
创建空 39
定义多个命令 39
更改 14
更新 36
管理 27, 40
恢复缺省 38
删除 37
使用输出模板恢复缺省值 38
示例 35
输出模板 28
输入模板 28
说明 18
为备用数据库生成 52
函数字符串继承 23
函数字符串类
rs_default_function_class 52
创建 23
创建, 基本 25
创建, 派生 24
更改主 Replication Server 25, 271
管理 23, 25
删除 27
说明 19
用于 DB2 数据库 11
指派给数据库 26
函数字符串效率 29, 35
函数作用域, 说明 13
恢复
RSSD 300
从 RSSD 故障中 298, 310
分区丢失或故障 290, 294
概述 298
从截断的主数据库日志中 294, 296
设置保存间隔 286
使用过程 283
脱机数据库日志中的消息 292
支持任务 310, 320
从主数据库故障中 296
主数据库和复制数据库 297
转储 290
转储中的 RSSD 299
恢复模式
Replication Server 312, 316
恢复守护程序 (dREC) 109
活动数据库 47
在切换后管理旧的活动数据库 81
重新启动客户端 81

I

ID Server
删除逻辑数据库 89
ignore loss 命令
处理丢失 316
忽略 SQM 和 DSI 丢失 316

和热备份应用程序 103
设置日志恢复后忽略 SQM 丢失 317
init_sqm_write_delay 配置参数 114
init_sqm_write_max_delay 配置参数 114
interfaces 文件
检查准确性 5
为热备份应用程序修改 84
isql 交互式 SQL 实用程序
验证服务器状态 5

J

基本函数字符串类
创建 25
集群
术语 345
计数器
SQM 命令高速缓存 139, 140
查看 251
查看信息 261
概述 251
用于查看的命令 251
重置 262
计数器名称 253
记录
分配器状态 108
加密列
热备份 61
监控
Replication Server 5
分区百分比 9
监控 DSI, 用于重新同步数据库 323
减少复制定义
热备份 95
脚本
验证服务器状态 5
截断的数据库日志, 恢复 294
解除绑定对象
到复制路径 236
净更改数据库
显示 199

K

可视化状态监控 6
可疑预订 101
客户端应用程序
在切换活动数据库后重新启动 81
空函数字符串, 创建 39
跨平台转储和装载 298
块大小
更改 213
块大小, 设置 111

L

load database 命令 72
load transaction 命令 72
LTL 命令
高速缓存 137
例外日志
访问 278
例外处理 276
删除事务 280
显示事务 278
例外系统日志事务
查询 278
连接 370, 375
设置保存间隔 288
连接管理器守护程序 (dCM) 109
列出绑定和对象
到复制路径 238
路由
RSSD 恢复 310
设置保存间隔 287
逻辑连接
send standby_repdef_cols 配置参数 87
创建 64
配置保存间隔 94
配置实现队列保存间隔 94
逻辑连接的 send standby_repdef_cols 配置参数 87
逻辑路径
删除 234
删除元素 234
添加 233
逻辑数据库连接
删除 89

M

materialization_save_interval 配置参数
用于逻辑连接 86
md_sqm_write_request_limit 参数 143
md_sqm_write_request_limit 配置参数 117, 129
mem_reduce_malloc 配置参数 115
mem_thr_dsi 配置参数 115
mem_thr_exec 配置参数 115

mem_thr_sqt 配置参数 115
mem_warning_thr1 配置参数 115
mem_warning_thr2 配置参数 115
memory_control 配置参数 115
memory_limit 配置参数 116
monSQLRepActivity 监控表 193
monSQLRepMisses 监控表 193
mount 命令 67
move primary 命令 25, 271
Multi-path Replication 217
命令
    admin config 172
    alter connection 171
    configure replication server 172
    hareg 350
命令和配置参数
    用于专用路由 243
命令批处理
    用于非 ASE 服务器 40
模块
    概述 251
    事务传送 108
    说明 105
    消息传送 108
目标作用域和复制定义作用域的函数字符串, 差异
        34

N

none
    事务序列化方法 158
none 函数字符串输出模板 29, 45
nrm_thread 配置参数 117
number of send buffers 配置参数 231
内存分配 212
内存消耗量参数交互 206
内存消耗量控制 146
    DSI、EXEC、SQT 线程 147
    HVAR 204, 206
    监控线程信息 147
    内存管理统计信息 148
    内存阈值警告消息 146

O

online database 命令 73
OQID 提交堆栈 164

P

parallel_dsi 配置参数 129, 153
派生的函数字符串类
    创建 24
派生函数字符串类, 描述 23
配置
    稳定队列高速缓存参数 134
配置参数
    dsi_bulk_copy 112, 171, 172
    dsi_bulk_threshold 112, 171, 172
    dsi_row_count_validation 274
    dynamic_sql 194
    dynamic_sql_cache_management 194
    dynamic_sql_cache_size 194
    mem_thr_dst 147
    mem_thr_exec 147
    mem_thr_sqt 147
    mem_warning_thr1 146
    mem_warning_thr2 146
    memory_control 147
    rs_config 系统表 111
    stats_reset_rssd 256
    多个主复制路径 237
    影响性能 111
配置概述 321
配置路由, 设置保存间隔 288
配置数据库重新同步 321
    获取数据库的转储 323
    监控 DSI 线程信息 323
    将转储应用于要重新同步的数据库 324
    向 Replication Server 发送 dump database 标
        记 323
    向 Replication Server 发送 resync database 标
        记 322
    指示 Replication Server 跳过事务 321
配置文件 370, 375
批处理函数字符串中的命令 39
批量插入
    请参见 批量拷入支持
批量拷入支持
    多语句事务, 支持 172
    连接参数 171
    连接参数, 检查值 172
    连接参数, 设置值 171
    命令 171
    数据服务器接口 (DSI), 实现 171
    预订实现, 更改 172
批量实现
    在热备份应用程序中 101

Q

quiesce database ... to manifest_file 命令 67
启用复制标记 67
迁移 RSSD 298
请求存储过程 332
    设置 337
    实现的先决条件 333
缺省的分区分配机制 248
缺省的函数字符串, 恢复 38

R

RCL 命令 340
    abort switch 命令 80
    admin log_name 命令 266
    admin logical_status 命令 79, 83
    admin set_log_name 266
    admin set_log_name 命令 4
    admin sqm_readers 命令 83
    admin who, dsi 命令 83
    admin who, sqm 命令 83, 286
    allow connections 命令 316
    alter connection 命令 26, 88, 290
    alter function string 命令 36
    alter function 命令 340
    assign action 命令 272
    configure connection 命令 40, 88, 290
    create connection 命令 26
    create error class 命令 268
    create function string class 命令 25
    create function string 命令 35
    create logical connection 命令 64
    drop connection 命令 81
    drop error class 命令 270
    drop function string class 命令 27
    drop function string 命令 37
    ignore loss 命令 316, 317
    move primary 命令 25, 271
    rebuild queues 命令 311
    resume connection 73
    resume connection 命令 73, 277
    set log recovery 命令 317
    suspend connection 命令 277
    sysadmin dropldb 命令 90
    sysadmin restore_dsi_saved_segments 命令 288
    wait for create standby 命令 73
    wait for switch 命令 80
rebuild queues 命令 311
rec_daemon_sleep_time 配置参数 117, 141
refimp analyze 359
refimp config 358
refimp run 359
REP_HVAR_ASE 许可证 196
RepAgent
    错误日志消息 266
    多个连接 229
    多个路径 229
RepAgent 用户线程 106
RepAgent 执行程序线程效率 211
replicate minimal columns
    和 rs_default_fs 系统变量 44
    和非缺省函数字符串 44
replicate minimal columns 子句, 使用 44
replicate minimal columns, 与动态 SQL 一起使用 195
replicate_minimal_columns 配置参数 86
Replication Server
    标准错误 4
    处理丢失的消息 315
    错误日志 82, 264
    独立模式 292, 311, 313
    分区 9
    复制中的处理 110
    恢复模式 312, 316
    监控 5
    检查错误 3
    内部 105, 111
    日志恢复模式 316
    信息性消息 264
    验证系统是否正常工作 4
    验证状态 5
    重建稳定队列 311
    主 Replication Server 中的处理 106, 110
Replication Server 程序
    rs_subcmp 315
Replication Server 错误类
    参数 273
Replication Server 系统数据库 (RSSD)
    从故障中恢复 298
    更新数据库生成号 319
resume connection 命令 73, 277
resume route 命令 245
resync 标记, 不使用任何选项 322
resync 标记, 发送 322
resync 标记, 使用 init 命令 323
resync 标记, 使用 purge 指令 322

RMS 心跳功能 250
RMS 中的心跳功能, 使用 250
RPC 函数字符串输出模板 29
RS 用户线程 110
rs_batch_end 系统函数 15
rs_batch_start 系统函数 15
rs_begin 系统函数 15
rs_check_repl 系统函数 15
rs_commit 系统函数 15
rs_config 系统表
    配置参数 111
rs_datarow_for_writetext 系统函数 17
rs_db2_function_class, 说明 20
rs_default_function_class 52
    说明 20
rs_delete 系统函数 17
rs_delexception 存储过程 280
rs_diskaffinity 系统表 249
rs_dumpdb 系统函数 15, 290
rs_dumptran 系统函数 15, 290
rs_get_charset 系统函数 15
rs_get_lastcommit 系统函数 15
rs_get_sortorder 系统函数 15
rs_get_textptr 系统函数 17
rs_get_thread_seq 系统函数 15, 166
rs_get_thread_seq_noholdlock 系统函数 15, 167
rs_helpclass 存储过程 43
rs_helperror 存储过程 273
rs_helpexception 存储过程 278
rs_helpfstring 存储过程 43
rs_helpfunc 存储过程 43
rs_idnames 系统表
    删除逻辑数据库 89
rs_init program
    添加备用数据库 72
    添加热备份数据库 65
rs_init 程序的转储标记选项 70, 82
rs_init_erroractions 存储过程 270
rs_initialize_threads 系统函数 15, 166
rs_insert 系统函数 17
rs_iq_function_class, 说明 20
rs_marker 系统函数 15
rs_mk_rsids_consistent 存储过程 304
rs_mss_function_class, 说明 20
rs_non_blocking_commit 系统函数 15
rs_non_blocking_commit_flush 系统函数 16
rs_oracle_function_class, 说明 20
rs_raw_object_serialization 函数 16
rs_repl_off 系统函数 16
rs_repl_on 系统函数 16
rs_rollback 系统函数 16
rs_select 系统函数 17
    更新函数字符串 37
rs_select_with_lock 系统函数 17
    更新函数字符串 37
rs_set_ciphertext 系统函数 16
rs_set_deml_on_computed 系统函数 16
rs_set_isolation_level 函数字符串 155
rs_set_isolation_level 系统函数 16
rs_set_non_blocking_commit 系统函数 16
rs_set_proxy 函数 16
rs_sqlserver_error_class 错误类 268
rs_sqlserver_function_class 25
    说明 20
rs_statcounters 系统表 261
rs_subcmp 程序 102, 315
rs_textptr_init 系统函数 17
rs_thread_check_lock 系统函数 16
rs_triggers_reset 系统函数 16
rs_trunc_reset 系统函数 16
rs_trunc_set 系统函数 16
rs_truncate 函数 17
rs_update 系统函数 17
rs_update_threads 系统函数 16, 166
rs_usedb 系统函数 17
rs_writetext 系统函数 17
RSFEATURE_HQ_INCR_CMPL_ON 跟踪标志,
        启用完全增量编译 206
RSI 线程
    说明 109
RSI 用户线程 110
rsi_batch_size 配置参数 130
rsi_packet_size 配置参数 131
rsi_sync_interval 配置参数 131
RSSD 故障
    恢复 298, 310
热备份
    带引号的标识符 62
    多个复制路径 242
    多个复制路径, 切换活动 242
    复制定义 95
    加密列 61
    减少复制定义 95
    预订 95
    重新同步数据库 329
    主数据库复制 75

热备份, alter table 命令支持 95
热备份的 alter table 命令支持 95
热备份环境
    多路径复制 241
    替代连接 241
    替代逻辑连接 241
热备份应用程序
    比较方法 53
    复制的信息 52
    用于复制数据库 92
    关闭复制 63
    监控 82
    逻辑连接 49
    名称相同的表 60
    强制复制 DDL 命令 62
    切换到备用数据库 76
    切换到备用数据库的影响 78
    设置数据库 63, 87
    数据库 49
    数据库连接 49
    物理连接 49
    限制 50
    用于主数据库 90
日志恢复
    为数据库设置 316
    正在检测丢失 317
入站队列
    多个读取器线程 87
    显示读取器线程 83
入站队列直接复制 137

S

save_interval 配置参数 286
    用于逻辑连接 86
send standby 子句
    用于参数 98
    用于列 98
set function string class 子句 26
set log recovery 命令 316
set replication Transact-SQL 命令 62, 88
set triggers off Transact-SQL 命令 88
skip to resync 标记, 从 RepAgent 发送到 Replication Server 322
skip to resync 参数 321
skip transaction 子句 278
smp_enable 配置参数 117
sp_dboption Adaptive Server 命令 206
sp_helpcounter 命令系统过程 261
sp_reptostandby 系统过程 54, 73
sp_setreplicate 系统过程
    将存储过程标记为要复制 339
sp_setrepproc 系统过程 59
    标记热备份活动数据库中的存储过程 73
sp_setreptable 系统过程
    标记热备份活动数据库中的表 73
SQL 语句复制
    Adaptive Server 监控表 193
    monSQLRepActivity 监控表 193
    monSQLRepMisses 监控表 193
    Replication Server 拓扑, 影响 175
    RSSD 修改 192
    WS_SQLDML_REPLICATION 参数 186
    表复制定义 185
    产品版本和混合版本要求 193
    多路径复制, 绑定对象 237
    复制 SQLDML 子句 184
    解决的问题 189
    配置复制定义 184
    启用 177
    数据库复制定义 184
    限制 190
    行计数验证 186
    阈值设置 180
    增强功能 174
    自动更正 192
    作用域 187
SQL 语句复制作用域 187
SQM 命令高速缓存 141
    计数器 139, 140
sqm_async_seg_delete 配置参数 117, 129
sqm_cache_enable 配置参数 118
sqm_cache_size 配置参数 118
sqm_cmd_cache_size 配置参数 129
sqm_max_cmd_in_block 配置参数 130
sqm_page_size 配置参数 118
sqm_recover_segs 配置参数 118
sqm_write_flush 配置参数 118, 121
sqt_init_read_delay 配置参数 118
sqt_max_cache_size 配置参数 119, 152
sqt_max_read_delay 配置参数 119
sqt_prs_cache_size 配置参数 119
stats_reset_rssd 配置参数 256
sts_cachesize 配置参数 120
sts_full_cache 配置参数 120
sub_daemon_sleep_time 配置参数 120
sub_sqm_write_request_limit 配置参数 120

Sun Cluster HA 345, 346
参考 345
suspend connection 命令 277
suspend route 命令 244
switch active 命令
在预订取消实现期间 102
在预订实现期间 100
在原子实现期间 100
sysadmin dropldb 命令 90
sysadmin restore_dsi_saved_segments 命令 288
删除
ID Server 中的逻辑数据库 89
函数字符串 37
函数字符串类 27
例外日志中的事务 280
逻辑路径 234
逻辑路径中的元素 234
逻辑数据库连接 89
物理路径 233
用户定义的函数 341
生成
性能报告 262
性能分析 262
性能概述 262
声明语句, 在语言输出模板中使用 41
失败的事务
处理 277
解决过程 277
时间戳
qid 318
实现队列保存间隔
strict 设置 94
设置逻辑连接的 94
使用 dist_direct_cache_read 提高分配器线程读取效率 212
使用 dsi_command_prefetch 提高 DSI 效率 210
使用 exec_nrm_request_limit 提高 RepAgent 执行程序线程效率 211
使用 mem_reduce_malloc 改进内存分配 212
使用 nrm_thread 提高 RepAgent 执行程序线程效率 211
示例
DSI 丢失检测 315
SQM 丢失检测 314
热备份应用程序 76
事务
大 154
例外处理 276
失败的原因 276
时间戳 318
使用并行 DSI 线程处理 148
小 154
序列化方法 156
装载日志转储 318
事务传送 (TD) 模块 108
事务名称, 缺省 161
事务中的 ddl, Adaptive Server sp_dboption 参数 206
守护程序
版本 (dVERSION) 110
报警 (dAlarm) 109
恢复 (dREC) 109
连接管理器 (dCM) 109
说明 105
异步 I/O (dAIO) 109
预订重试 (dSUB) 109
杂项 109
输出模板 18
none 29, 30, 35, 45
rpc 29
writetext 45
创建空函数字符串 39
恢复缺省函数字符串 38
语言 29
输入模板 18
输入模板, 示例 31
数据服务
将 Replication Server 作为 350
启动/关闭 350
数据服务器
错误处理 267, 273
数据服务器接口 171, 172
数据库
备用 49
故障 296
活动 49
逻辑 49
设置日志恢复 316
指派函数字符串类 26
自定义操作 11, 43
数据库的转储, 获取 323
数据库的转储, 应用 324
数据库连接
并行 DSI
参数 149
配置参数
用于并行 DSI 149
用于热备份应用程序 49

为并行 DSI 配置 149
数据库日志
截断的主数据库恢复 294
联机恢复消息 294
脱机恢复消息 292
为重装确定 318
重新装载 319
数据库生成号
qid 319
在数据库恢复期间调整 318
和转储 319
数据库重新生成号, 重置 320
数据库重新同步
多路径复制, 绑定对象 237
数据库重新同步方案 324
使用第三方转储实用程序重新同步 326
通过相同的转储来重新同步主数据库和复制数据库 328
无 resync database 标记支持时的重新同步 327
直接从主数据库重新同步复制数据库 324
重新同步热备份应用程序中的活动数据库和备用数据库 329
数据类型
text 和 image 53
刷新的值
查看 259
死锁检测, 并行 dsi 165

T

tempdb, 在 Adaptive Server 中 206
Transact-SQL 命令
dump database 290
dump transaction 290
set replication off 88
set triggers off 88
truncate table 命令 281
RCL 53
复制 87
提高的 DSI 效率 210
提高的 RepAgent 执行程序线程效率 211
提高的分配器线程读取效率 212
提示 249
替代复制连接
创建 223
更改 224
示例, 创建 225
显示 224
预订, 移动 228
替代连接
创建预订 227
预订, 创建 227
替代主连接
显示 227
添加
逻辑路径 233
物理路径 232

U

use_batch_markers 配置参数 130
USER 线程 110

W

wait for create standby 命令 73
wait for switch 命令 80
writetext 函数字符串输出模板 45
完全增量编译
Adaptive Server, 启用 206
RSFEATURE_HQ_INCR_CMPL_ON 跟踪标志 206
用于 HVAR 205
完全增量编译的跟踪标志 206
为非 Sybase 复制数据服务器设置隔离级别 155, 169
维护用户
用于备用数据库 74
添加 72
文件
Replication Server 错误日志 3
标准错误输出 4
稳定队列 121
DSI 丢失 313
处理分区故障 288
从数据库日志中脱机重建 312
高速缓存 134
检测丢失 313
联机重建 311
重建 311
稳定队列管理器线程 (SQM)
处理丢失 315
日志恢复后进行丢失检测 317
在稳定队列重建过程中检测到丢失 313
稳定队列事务 (SQT) 线程 107
无 resync database 标记支持
重新同步数据库 327

物理路径
删除 233
添加 232

X

xpdl 298
系统表
rs_diskaffinity 249
rs_idnames 89
rs_statcounters 261
系统过程
sp_helpcounter 命令 261
sp_setreplicate 339
sp_setrepproc 73
sp_setreptable 73
系统函数
rs_dumpdb 290
rs_dumptran 290
说明 12
系统函数, 列表
具有复制定义作用域 17
具有函数字符串类作用域 14
系统事务 281
显示
rs_helpclass 存储过程 271
错误类信息 271
例外日志中的事务 278
为错误号指定的操作 273
与函数相关的信息 42
专用路由信息 245
线程
DSI 调度程序 108, 153
DSI 执行程序 108, 153
RS 用户 110
RSI 109
RSI 用户 110
USER 110
用于并行 DSI 148
分配器 (dist) 107
说明的主 Replication Server 中 106, 110
为复制系统显示 6
稳定队列管理器 (SQM) 106
稳定队列事务 (SQT) 106
线程, 杂项 109
限制
热备份应用程序 50
消息
SQM 丢失检测 317
处理稳定队列中的丢失 315
从联机数据库日志中恢复 294
从脱机数据库日志中恢复 292
消息传送 (MD) 模块 108
小事务 154
协调的转储
创建 290
恢复数据库 296
装载主数据库和复制数据库 297
写操作 121
新增功能
Replication Server 15.1 ESD #1 171
信息性消息
格式 264
行计数验证
非 SQL 语句复制 274
禁用 274
示例 273
显示表名 275
增强功能 274, 275
行计数验证, 在 SQL 语句复制中 186
修饰符
函数字符串 32
在函数字符串变量中 31
序列化方法
no_wait 157
none 157
wait_for_commit 158
wait_for_start 157

Y

严重级
Replication Server 272
错误消息 265
数据服务器错误 272
一致性
为复制数据库保持 290
移动预订 228
异步 I/O 守护程序 (dAIO) 109
异步存储过程
和非唯一的用户定义的函数名称 343
将参数添加到 340
请求存储过程 332
应用的 332
用户定义的函数 339
执行 331
应用存储过程
设置 334

实现的先决条件 333
用户定义的函数
关联复制的存储过程 339
管理 339
删除 341
说明 13
添加参数 340
映射到不同的存储过程 342
指定非唯一函数名称 343
用于热备份的主数据库复制
设置 75
语言
函数字符串输出模板 29
预订
备份后重新创建 308
恢复备份之后比较 302
热备份数据库中的数据 99
热备份应用程序中的限制 99
预订分析引擎 (SRE) 107
预订迁移
说明 107
预订实现
预订重试守护程序 (dSUB) 109
阈值, 在 SQL 语句复制中设置 180
阈值级别
为分区设置和使用 9
元数据减少, 用于表 133
原始队列 ID (qid) 318
确定数据库生成号 319
原子实现
在热备份应用程序中 100
约定
样式 1
语法 1

Z

增加队列块大小 212
增强的内存分配 212
增强功能
Replication Server 性能 171
SQL 语句复制 174
支持
请参见 批量拷入支持
执行程序命令高速缓存 133
表元数据减少 133
大小, 设置 134
直接 I/O 121
直接命令复制
出站队列 139
入站队列 137
直接写入介质 121
重试机制, 为 HVAR 增强 204
重新开始连接, 使用 skip to resync 标记 321
重新生成号, 重置 320
重新同步 Adaptive Server 数据库
简介 321
支持的 Adaptive Server 和 RepAgent 版本 321
重新同步数据库 321
resync 标记, 发送 322
skip to resync 参数 321
发送 dump database 标记 323
方案 324
方案, 热备份 329
方案, 无 resync database 标记支持 327
获取数据库转储 323
监控 DSI 323
配置 321
使用 skip to resync 参数重新开始连接 321
跳过事务 321, 322
应用数据库的转储 324
重置数据库生成号 320
主 Replication Server
处理 106, 110
更改错误类的 271
将函数字符串类更改为另一个 Replication Server 25
主键
用于热备份数据库中的表 97
主节点
为错误类指定 269
主连接
替代, 显示 227
主数据库
DDL 命令和系统过程 57
从故障中恢复 296
从转储装载 296
复制 74
复制限制 58
恢复截断的日志 294
和热备份应用程序 51
主转储
恢复主数据库 296
专用路由 243
创建 243
命令和配置参数 243
转储
初始化热备份数据库 67, 72

创建 290
事务时间戳 318
数据库生成号 319
为重装确定 318
装载
转储中的主数据库 296
状态
监控 6
验证 RepAgent 5
验证 Replication Server 5
验证数据服务器 5
状态监控 6
作用域, 函数的 13