SQL Remote™

2009 年 2 月

11.0.1 版

版权和商标

目录

关于本手册

主题

本手册介绍用于移动计算的 SQL Remote 数据复制系统，此系统支持使用电子邮件或文件传输等间接链接共享 SQL Anywhere 统一数据库和多个 SQL Anywhere 远程数据库之间的数据。

目标读者

本书面向希望在其信息系统中添加 SQL Remote 复制的 SQL Anywhere 用户。

关于 SQL Anywhere 文档

完整的 SQL Anywhere 文档以四种形式提供，但所包含信息均相同。

- **HTML 帮助** 联机帮助文档包含完整的 SQL Anywhere 文档，其中包括手册和 SQL Anywhere 工具的上下文相关帮助。

  如果使用 Microsoft Windows 操作系统，则联机帮助文档以 HTML 帮助 (CHM) 格式提供。若要访问此文档，请选择 [开始] » [程序] » [**SQL Anywhere 11**] » [文档] » [联机手册]。

  管理工具使用同一联机文档来实现帮助功能。

- **Eclipse** 在 Unix 平台上以 Eclipse 格式提供完整的联机帮助。要访问文档，请从 SQL Anywhere 11 安装的 *bin32* 或 *bin64* 目录下运行 *sadoc*。

- **DocCommentXchange** DocCommentXchange 是一个用于访问和讨论 SQL Anywhere 文档的社区。

  使用 DocCommentXchange 可以执行以下任务：

  - 查看文档
  - 检查是否有用户对文档各部分所做出的阐明
  - 提供建议和修正意见以在将来的版本中为所有用户改进文档

  访问 http://dcx.sybase.com。

- **PDF** 整套 SQL Anywhere 手册会以一组 Portable Document Format (PDF) 文件的形式提供。您必须有 PDF 阅读器才能查看信息。要下载 Adobe Reader，请访问 http://get.adobe.com/reader/。

  若要在 Microsoft Windows 操作系统上访问 PDF 文档，请选择 [开始] » [程序] » [**SQL Anywhere 11**] » 文档 » [联机手册 - PDF 格式]。

  要在 Unix 操作系统上访问 PDF 文档，请使用 Web 浏览器打开 *install-dir/documentation/zh/pdf/index.html*。

关于文档集中的手册

SQL Anywhere 文档由以下手册组成：

- **SQL Anywhere 11 - 简介** 本手册介绍 SQL Anywhere 11，一个提供数据管理和数据交换技术的综合数据包，通过它可以为服务器环境、台式机环境、移动环境以及远程办公环境快速开发由数据库驱动的应用程序。
- **SQL Anywhere 11 - 更改和升级** 本手册介绍 SQL Anywhere 11 以及该软件以前版本中的新功能。
- **SQL Anywhere 服务器 - 数据库管理** 本手册介绍如何运行、管理及配置 SQL Anywhere 数据库。它介绍了数据库连接、数据库服务器、数据库文件、备份过程、安全性、高可用性、使用复制服务器进行复制以及管理实用程序和选项。

- **SQL Anywhere 服务器 - 编程** 本手册介绍如何使用 C、C++、Java、PHP、Perl、Python 和 .NET 编程语言（例如 Visual Basic 和 Visual C#）建立和部署数据库应用程序。其中介绍了各种编程接口，如 ADO.NET 和 ODBC。
- **SQL Anywhere 服务器 - SQL 参考** 本手册提供了系统过程和目录（系统表和视图）的参考信息。也介绍了 SQL 语言（搜索条件、语法、数据类型和函数）的 SQL Anywhere 实现。
- **SQL Anywhere 服务器 - SQL 的用法** 本手册介绍如何设计和创建数据库；如何导入、导出和修改数据；如何检索数据以及如何建立存储过程和触发器。
- **MobiLink - 入门** 本手册介绍基于会话的关系数据库同步系统 MobiLink。MobiLink 技术支持双向复制并且非常适用于移动计算环境。
- **MobiLink - 客户端管理** 本手册介绍如何设置、配置和同步 MobiLink 客户端。MobiLink 客户端可以是 SQL Anywhere 或者 UltraLite 数据库。本手册同时也介绍了 Dbmlsync API，通过它可无缝地将同步集成到 C++ 或 .NET 客户端应用程序中。
- **MobiLink - 服务器管理** 本手册说明如何设置和管理 MobiLink 应用程序。
- **MobiLink - 服务器启动的同步** 本手册介绍 MobiLink 服务器启动的同步，这种功能允许 MobiLink 服务器启动同步或在远程设备上进行操作。
- **QAnywhere** 本手册介绍 QAnywhere，一个用于移动、无线、台式机和膝上型客户端的消息传递平台。
- **SQL Remote** 本手册介绍用于移动计算的 SQL Remote 数据复制系统，此系统支持使用电子邮件或文件传输等间接链接共享 SQL Anywhere 统一数据库和多个 SQL Anywhere 远程数据库之间的数据。
- **UltraLite - 数据库管理和参考** 本手册介绍适用于小型设备的 UltraLite 数据库系统。
- **UltraLite - C 及 C++ 编程** 本手册介绍 UltraLite C 和 C++ 编程接口。利用 UltraLite，可以开发数据库应用程序，并将它们部署到手持式设备、移动设备或嵌入式设备。
- **UltraLite - M-Business Anywhere 编程** 本手册介绍 UltraLite for M-Business Anywhere。利用 UltraLite for M-Business Anywhere，用户可以开发基于 Web 的数据库应用程序，并将它们部署到运行 Palm OS、Windows Mobile 或 Windows 的手持式设备、移动设备或嵌入式设备。
- **UltraLite - .NET 编程** 本手册介绍 UltraLite.NET。利用 UltraLite.NET，您可以开发数据库应用程序，并将它们部署到计算机、手持式设备、移动设备或嵌入式设备。
- **UltraLiteJ** 本手册介绍 UltraLiteJ。利用 UltraLiteJ，可以在支持 Java 的环境中开发和部署数据库应用程序。UltraLiteJ 支持 BlackBerry 智能手机和 Java SE 环境。UltraLiteJ 基于 iAnywhere UltraLite 数据库产品。
- **错误消息** 本手册提供了 SQL Anywhere 错误消息及其诊断信息的完整列表。

文档约定

本节列出了本文档中使用的约定。

操作系统

SQL Anywhere 可以在各种平台上运行。在大多数情况下，该软件在所有平台上的行为都是相同的，但也有变动或限制。这些变动或限制通常基于基础操作系统（Windows、Unix），很少基于特定变型（AIX、Windows Mobile）或版本。

为了简化对操作系统的提及，本文档按如下方式对支持的操作系统进行分组：

- **Windows** Microsoft Windows 系列包括 Windows Vista 和 Windows XP（主要用于服务器、台式计算机和膝上型计算机），以及 Windows Mobile（用于移动设备）。

  除非另外指定，否则当本文档提及 Windows 时，是指所有基于 Windows 的平台，包括 Windows Mobile。

- **Unix** 除非另外指定，否则当本文档提及 Unix 时，是指所有基于 Unix 的平台，包括 Linux 和 Mac OS X。

目录和文件名

大部分情况下，对目录和文件名的引用在所有支持的平台上都是类似的，只需在不同形式之间进行简单的转换。这时需使用 Windows 约定。在细节更为复杂的情况下，文档显示所有相关形式。

下面是文档编写中用于简化目录和文件名的约定：

- **大写和小写目录名** 在 Windows 和 Unix 上，目录和文件名可以包括大写和小写字母。创建目录和文件时，文件系统会保留字母大小写。

  在 Windows 上，对目录和文件的提及*不区分大小写*。混合使用大小写的目录和文件名很常见，但使用所有小写字母来提及目录和文件的形式也很常见。SQL Anywhere 安装包含诸如 *Bin32* 和 *Documentation* 的目录。

  在 Unix 上，对目录和文件的提及*区分大小写*。混合使用大小写的目录和文件名不常见。大多数的目录和文件名全部使用小写字母。SQL Anywhere 安装包含诸如 *bin32* 和 *documentation* 的目录。

  本文档采用 Windows 形式的目录名。大多数情况下，在 Unix 上可以将大小写混合形式的目录名转换成小写字母的等效目录名。

- **分隔目录和文件名的斜线** 文档使用反斜线作为目录分隔符。例如，PDF 格式的文档位于 *install-dir\Documentation\zh\PDF*（Windows 形式）。

  在 Unix 上，用正斜线替换反斜线。PDF 文档位于 *install-dir/documentation/zh/pdf* 下。

- **可执行文件** 文档使用 Windows 约定显示可执行文件名（带有诸如 *.exe* 或 *.bat* 后缀）。在 Unix 上，可执行文件名没有后缀。

  例如，在 Windows 上，网络数据库服务器是 *dbsrv11.exe*。在 Unix 上是 *dbsrv11*。

- ***install-dir*** 在安装过程中，选择 SQL Anywhere 的安装位置。创建环境变量 SQLANY11，用来表示此位置。文档中以 *install-dir* 表示此位置。

  例如，本文档将此文件表示为 *install-dir*\readme.txt。在 Windows 上，这等同于 *%SQLANY11%\readme.txt*。在 Unix 上，这等同于 *$SQLANY11/readme.txt* 或 *${SQLANY11}/readme.txt*。

有关 *install-dir* 缺省位置的详细信息，请参见“SQLANY11 环境变量”一节 《SQL Anywhere 服务器 - 数据库管理》。

- ***samples-dir*** 在安装过程中，选择 SQL Anywhere 随附的示例的安装位置。创建环境变量 SQLANYSAMP11，用来表示此位置。文档中以 *samples-dir* 表示此位置。

  要在 *samples-dir* 中打开 Windows 资源管理器窗口，请在 [**开始**] 菜单中，选择 [**程序**] » [**SQL Anywhere 11**] » [**示例应用程序和项目**]。

  有关 *samples-dir* 缺省位置的详细信息，请参见“SQLANYSAMP11 环境变量”一节 《SQL Anywhere 服务器 - 数据库管理》。

命令提示符和命令 shell 语法

大多数操作系统都提供一种或多种使用命令 shell 或命令提示符来输入命令和参数的方法。Windows 命令提示符包括 Command Prompt（DOS 提示符）和 4NT。Unix 命令 shell 包括 Korn shell 和 bash。每个 shell 都具有一些功能，其能力不仅仅局限于简单命令。这些功能通过特殊字符来驱动。特殊字符和功能随 shell 的不同而不同。如果没有正确使用这些特殊字符，通常会导致语法错误或意外行为。

本文档以普通形式提供命令行示例。如果这些示例中包含 shell 的特殊字符，则命令需要根据特定 shell 进行修改。修改方法不在本文档所述范围之内，但通常是在包含这些特殊字符的参数两旁加上引号，或是在特殊字符前面使用转义字符。

下面是命令行语法的一些示例，不同的平台可能会有不同的形式：

- **括号和大括号** 有些命令行选项需要一个参数，该参数将以列表形式接受详细的值指定。该列表通常用括号或大括号括起来。本文档使用括号。例如：

  ```
  -x tcpip(host=127.0.0.1)
  ```

  如果括号导致出现语法问题，用大括号替代：

  ```
  -x tcpip{host=127.0.0.1}
  ```

  如果两种形式都将产生语法问题，应按照 shell 的要求，用引号将整个参数括起来：

  ```
  -x "tcpip(host=127.0.0.1)"
  ```

- **引号** 如果必须在参数值中指定引号，该引号可能会与用于括参数的引号的传统用法发生冲突。例如，要指定值中包含双引号的加密密钥，则可能必须用引号括起密钥，然后转义嵌入的引号：

  ```
  -ek "my \"secret\" key"
  ```

  在许多 shell 中，密钥的值为 `my "secret" key`。

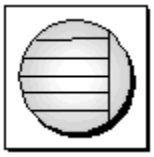
- **环境变量** 本文档介绍设置环境变量。在 Windows shell 中，环境变量使用语法 *%ENVVAR%* 来指定。在 Unix shell 中，环境变量使用语法 *$ENVVAR* 或 *${ENVVAR}* 来指定。

图标

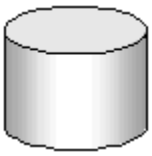
本文档中使用了下列图标。

- 客户端应用程序。

- 数据库服务器，如 Sybase SQL Anywhere。



- 数据库。在某些高水平的图中，可以使用此图标表示数据库和管理该数据库的数据库服务器。



- 复制或同步中间件。用于帮助在数据库之间共享数据。例如 MobiLink 服务器和 SQL Remote 消息代理。

- 编程接口。

联系文档小组

我们欢迎您就本帮助文档提出意见、建议和反馈信息。

要提交意见和建议，请发送电子邮件到 SQL Anywhere 文档小组，地址为 iasdoc@sybase.com。虽然我们不对这些电子邮件进行回复，但您的反馈会帮助我们改进文档，因此我们真诚地欢迎您提出宝贵的意见和建议。

DocCommentXchange

也可以使用 DocCommentXchange 将意见或建议直接置于帮助主题中。DocCommentXchange (DCX) 是一个用于访问和讨论 SQL Anywhere 文档的社区。使用 DocCommentXchange 可以执行以下任务：

- 查看文档
- 检查是否有用户对文档各部分所做出的阐明
- 提供建议和修正意见以在将来的版本中为所有用户改进文档

访问 http://dcx.sybase.com。

查找详细信息并请求技术支持

附加信息和资源可从 Sybase iAnywhere 开发人员社区获得，网址是 http://www.sybase.com/developer/library/sql-anywhere-techcorner。

如果您有问题或是需要帮助，可将邮件发布到下面所列的 Sybase iAnywhere 新闻组。

当您向这些新闻组发布邮件时，请务必提供问题的详细信息，包括 SQL Anywhere 版本的内部版本号。可以通过运行以下命令找到此信息：**dbeng11 -v**。 **dbeng11 -v**.

新闻组位于 *forums.sybase.com* 新闻服务器上。

这些新闻组包括：

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

有关 Web 开发问题，请访问 http://groups.google.com/group/sql-anywhere-web-development。

新闻组免责声明

iAnywhere Solutions 没有义务为其新闻组提供解决方案、信息或建议，除提供系统操作员监控服务和确保新闻组的运行和可用性外，iAnywhere Solutions 也没有义务提供任何其它服务。

如果时间允许，iAnywhere 技术顾问以及其他员工也会对新闻组服务提供帮助。他们是在自愿的基础上提供帮助的，所以可能无法定期提供解决方案和信息。他们可以提供多少帮助取决于他们的工作量。

SQL Remote 复制设计

本节将提供有关安装 SQL Remote 复制项的概述和说明。

SQL Remote 简介

目录

SQL Remote 的功能

SQL Remote 是一种基于消息的技术，是为在统一数据库和大量远程数据库之间双向地复制数据库事务而设计的。远程站点的管理和资源要求已降到了最低限度，从而使 SQL Remote 能够很好地适用于移动设备。

SQL Remote 提供以下功能：

- **支持多个预定者** SQL Remote 允许偶尔连接的用户在 SQL Anywhere 统一数据库和大量 SQL Anywhere 远程数据库（通常包括许多移动数据库）之间复制数据。
- **基于事务日志的复制** SQL Remote 使用事务日志进行复制。因此，在进行更新时只复制经过更改的数据。这确保了整个复制系统中适当的事务原子性，同时保持复制中所涉及的数据库之间的一致性。
- **集中管理** 在统一数据库上集中管理 SQL Remote。一个公司可以拥有数量庞大的移动工作人员和许多各不相同的数据库，但每个远程数据库并不需要逐个维护。此外，SQL Remote 操作对最终用户是不可见的。
- **节约内存** 为了高效率地运行，SQL Remote 使用内存比较节省。这允许您在现有远程计算机和设备上使用 SQL Remote 而不必在新硬件上进行投资。可以在空间有限的远程计算机和设备上进行传入和传出复制；只有相关的数据才会从统一数据库复制到远程数据库。
- **支持多种平台** 多个操作系统和消息链接都支持 SQL Remote。SQL Anywhere 数据库可从一个文件或操作系统复制到另一个文件或操作系统。 “支持的平台”一节 《SQL Anywhere 11 - 简介》.

另请参见

- “比较同步技术”一节 《SQL Anywhere 11 - 简介》
- “选择同步技术”一节 《SQL Anywhere 11 - 简介》

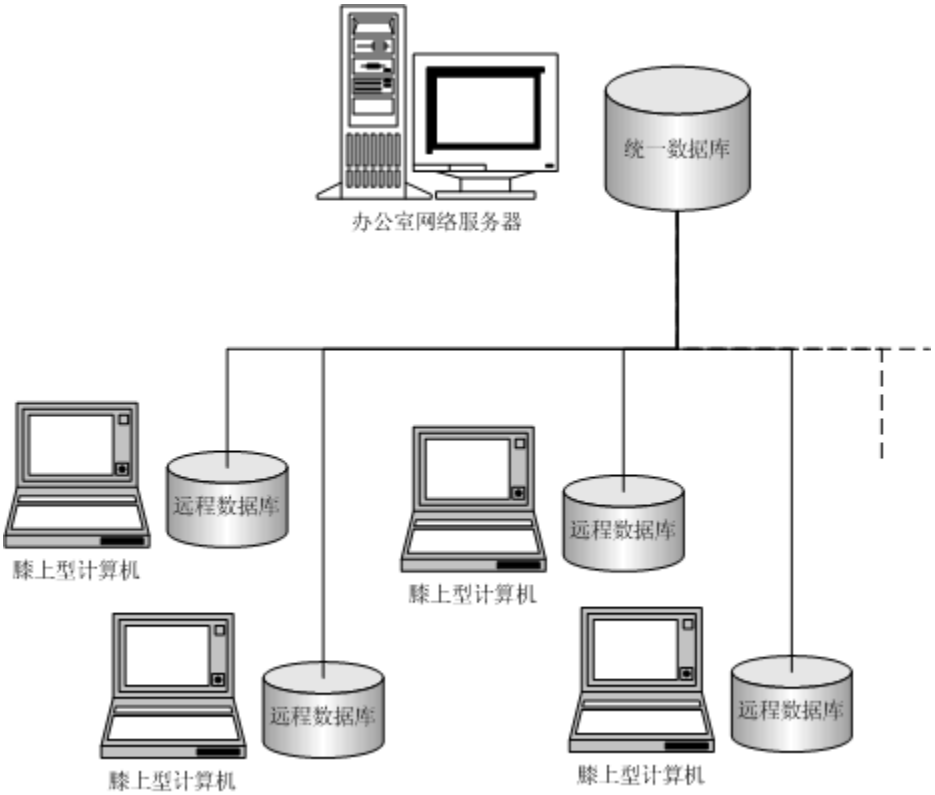
典型 SQL Remote 设置

SQL Remote 是专为具有以下要求的复制系统设计的：

- **大量远程数据库** 因为许多远程数据库的消息可以同时准备，所以 SQL Remote 可以在单个安装中支持数以千计的远程数据库。
- **不定时连接** SQL Remote 支持不定时连接到或间接连接到网络的数据库。SQL Remote 无法实现数据在各个站点的即时可用性。例如，它可以使用 SMTP 电子邮件系统传送复制。
- **短到长等待时间** 长等待时间意味着，在一个数据库中输入数据后到将数据复制到系统中的每个数据库之间的延迟时间非常长。使用 SQL Remote，可按几秒钟、几分钟、几小时或几天的时间间隔发送复制消息。
- **低到中等容量** 由于复制消息是不定时传递的，所以如果每个远程数据库的事务量很大，则会导致消息量也很大。SQL Remote 最适于每个远程数据库的复制数据量相对较小的系统。在统一数据库上，SQL Remote 可同时为多个数据库准备消息。
- **异类数据库** 系统中的每个 SQL Anywhere 数据库都必须具有相似的模式。

适用于移动工作人员的服务器到远程的数据库复制

在下面示例中，SQL Remote 提供了办公室网络上的统一数据库与销售代表所用膝上型计算机上的个人数据库之间的双向复制。SMTP 电子邮件系统用作一种消息传送手段。



为了管理统一数据库，办公室网络服务器运行 SQL Anywhere 数据库服务器。SQL Remote 连接统一数据库的方式与其它任何客户端应用程序的连接方式相同。

每个销售代表的膝上型计算机上都包括 SQL Anywhere 个人服务器、SQL Anywhere 远程数据库和 SQL Remote。

离开办公室时，销售代表可以连接到 Internet 上以运行 SQL Remote，执行以下功能：

- 收集来自办公室网络服务器上的统一数据库上的发布更新。
- 将所有本地更新（如新订单）提交到办公室网络服务器上的统一数据库。

来自办公室网络数据库的发布更新可能包括需要销售代表处理的产品的新特殊情况、新的定价和库存信息。膝上型计算机上的 SQL Remote 读取这些更新，并自动将其应用到销售代表的远程数据库中，这一过程中不需要销售代表进行任何额外操作。

销售代表记录的新订单也将自动提交并应用到办公室网络数据库，而无需销售代表执行任何额外操作。

办公室之间的服务器到服务器数据库复制

在此示例中，SQL Remote 提供了销售办公室或发售部门的数据库服务器与总公司办公室的数据库服务器之间的双向复制。在销售办公室仅需要的操作是初始设置和对服务器的持续维护。

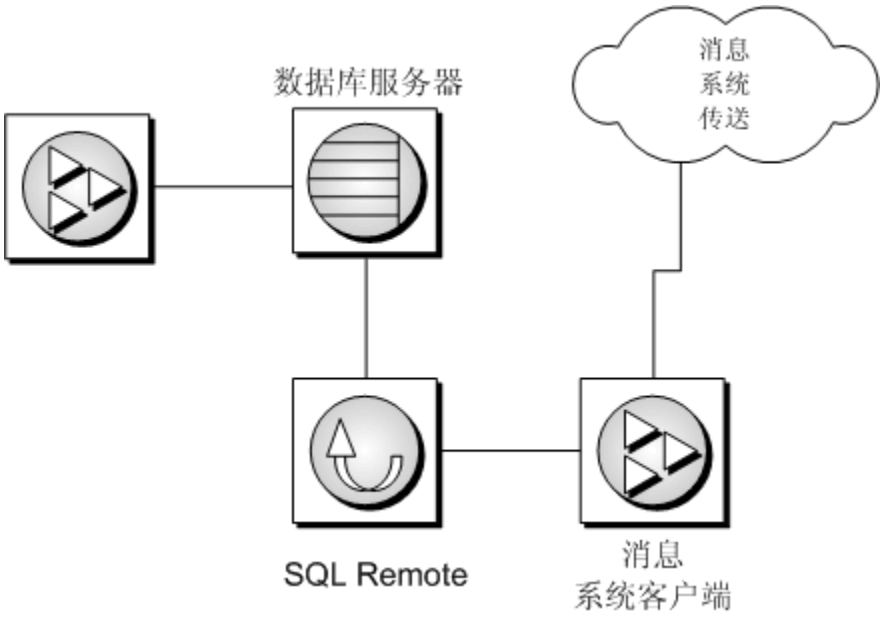
可以向 SQL Remote 层次中添加多个层：例如，每个销售办公室服务器都可作为一个统一数据库，支持为该办公室工作的远程预定者。

膝上型计算机

可将 SQL Remote 配置为允许每个办公室接收自己的数据集。员工记录之类的表可以在数据库中保持其私密性，这个数据库与复制的数据所在的数据库相同。

SQL Remote 组件

下面是 SQL Remote 所必需的组件：



- **数据库服务器** 统一站点和每个远程站点上都需要 SQL Anywhere 数据库。
- **SQL Remote** 要发送和接收数据库到数据库的复制消息，必须在统一站点和远程站点上安装 SQL Anywhere。

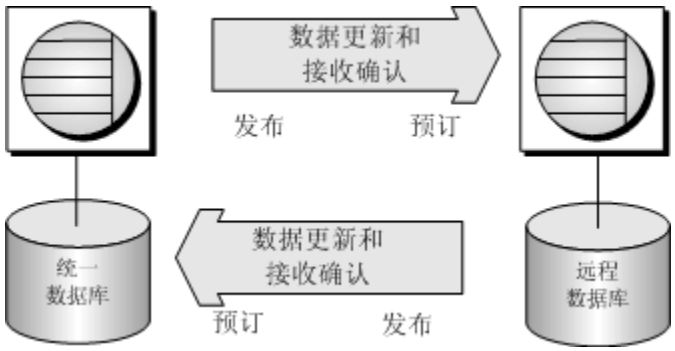
  SQL Remote 消息代理连接到数据库服务器的方式是通过客户端/服务器连接。消息代理可以与数据库服务器在同一台计算机上运行，也可以不在同一台计算机上运行。
- **消息系统客户软件** SQL Remote 使用现有的消息系统传送复制消息。

  如果使用的是共享文件或 FTP 消息系统，会随操作系统提供该消息系统。

  如果使用的是 SMTP 电子邮件系统，那么在统一站点和每个远程站点都必须安装电子邮件客户端。
- **客户端应用程序** 客户端应用程序可以使用 ODBC、嵌入式 SQL 或其它几种编程接口。客户端应用程序无需知道它们使用的是统一数据库还是远程数据库。从客户端应用程序的角度来看，它们没有区别。请参见“SQL Anywhere 数据访问编程接口” 《SQL Anywhere 服务器 - 编程》。

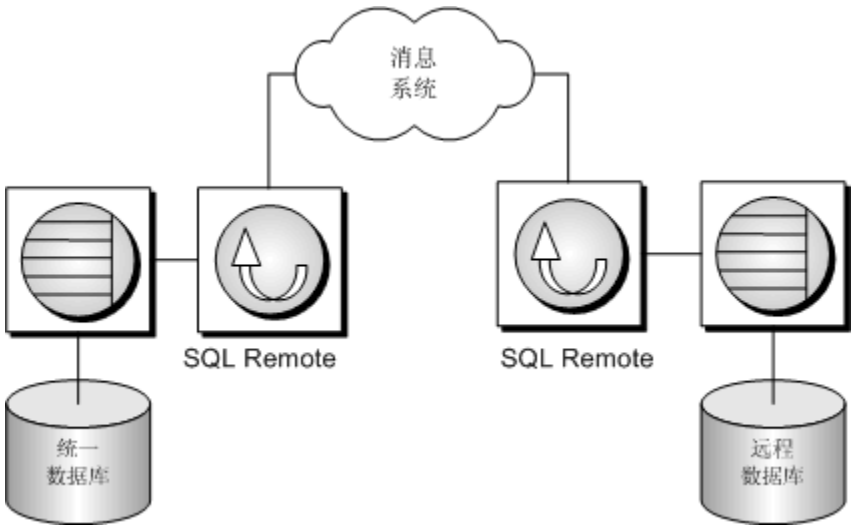
了解 SQL Remote 复制过程

使用 SQL Remote，消息总是双向发送。统一数据库将包含发布更新的消息发送到远程数据库，远程数据库也将更新的数据和收到确认消息发送到统一数据库。



在远程数据库用户修改数据时，他们进行的更改也被复制到统一数据库。当在统一数据库上应用这些更改时，这些更改即成为统一数据库发布的一部分，并被包括在发送到所有远程数据库（更新内容所来自的远程数据库除外）的更新中。利用这种方法，通过统一数据库实现远程数据库之间的复制。

例如，如果统一数据库上发布中的数据进行了更新，这些更新会被发送到远程数据库。即使远程数据库上的数据从未更新过，确认消息仍被发送回统一数据库，以跟踪复制的状态。



♦ **SQL Remote 复制过程所涉及的步骤**

1. 在每个参与复制的统一数据库和远程数据库上，都有管理复制的消息代理和事务日志。所有已提交的更改都被记录下来，并保存在事务日志中。

2. 统一数据库上的消息代理将定期地扫描事务日志，并将所有已提交的对每个发布（数据部分）进行的事务打包成消息。然后，统一数据库的消息代理将相关的更改发送到预订这些发布的远程用户。消息代理使用消息传递系统发送更改。SQL Remote 支持 SMTP 电子邮件系统、FTP 和 FILE。
3. 远程数据库上的消息代理接收由统一数据库发来的消息，并向统一数据库发送一个消息已收到的确认。然后，消息代理将事务应用到远程数据库。
4. 远程用户可随时运行消息代理，将在远程数据库上进行的事务打包成消息，然后发回统一数据库。
5. 统一站点上的消息代理会处理发自远程数据库的消息，并将事务应用到统一数据库。

SQL Remote 复制设计和设置

目录

创建 SQL Remote 系统

使用统一数据库完成所有 SQL Remote 管理任务。

♦ 创建 SQL Remote 系统

1. 选择 SQL Anywhere 统一数据库或创建新的 SQL Anywhere 数据库。远程数据库（也就是 SQL Anywhere 数据库）是从统一数据库创建的。

   创建新的 SQL Anywhere 数据库时，应考虑 SQL Remote 如何使用主键。例如，最好选择带有全局自动增量的 BIGINT 作为主键列的数据类型。请参见“重复主键错误”一节第 53 页。

2. 确定要复制的数据。

   创建有效的复制系统时，需要确定要使用的表以及表中的列，最后还要确定要复制的行的子集。只包括所需的信息。

3. 在统一数据库中创建发布。

   SQL Remote 使用发布和预订模型来确保正确的信息到达目标用户。在统一数据库中将要复制的数据安排到发布中。请参见“发布和项目”一节第 13 页。

4. 在统一数据库中创建发布者用户。

   发布者是具有 PUBLISH 权限的用户，用于唯一标识统一数据库。请参见“PUBLISH 权限”一节第 23 页。

5. 创建远程用户。

   远程用户用于唯一标识远程数据库。请参见“REMOTE 权限”一节第 25 页。

   创建远程用户时，可定义在传输数据时要使用的消息类型，也可以定义发送数据的频率。

6. 通过创建预订为远程用户预订发布。请参见“预订”一节第 33 页。

7. 确定远程用户可以使用数据的方式。

   远程用户始终都可以读取数据。也可以允许他们更新、删除和插入数据。请参见“了解基于事务日志的复制”一节第 35 页。

8. 选择解决冲突的方法。

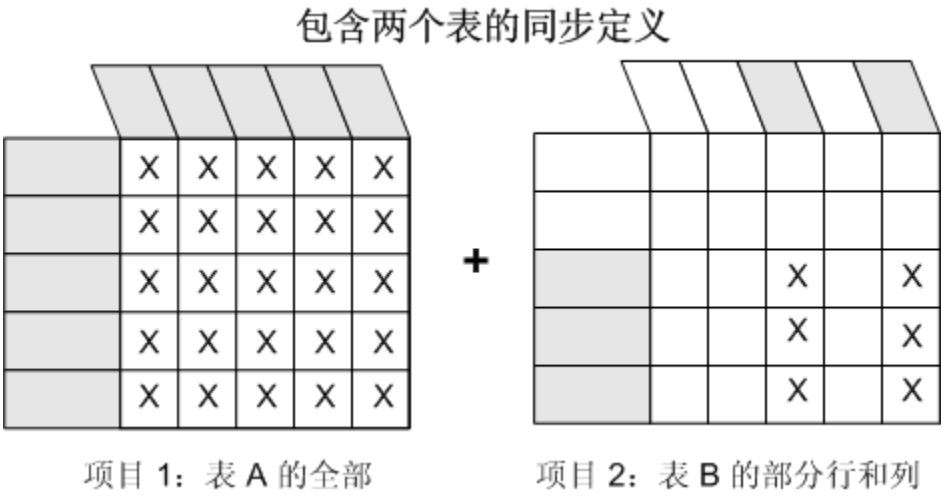
   当远程用户更新、删除或插入数据时，可能会在复制过程中发生冲突。必须实施各方法来解决冲突。请参见“更新冲突的缺省解决方法”一节第 43 页。

9. 部署 SQL Remote 系统。

   创建远程数据库并安装相应的软件。请参见“SQL Remote 的管理概述”一节第 78 页。

发布和项目

发布用于定义要复制的数据集。发布可包括来自多个数据库表的数据。**项目**指的是发布中的表。发布中的每个项目可由整个表构成，也可由表中行和列的子集构成。



限制

发布不能包括视图或存储过程。有关 SQL Remote 如何复制过程和触发器的信息，请参见“复制过程”一节第 38 页和“复制触发器”一节第 38 页。

查看发布和项目 (Sybase Central)

在 Sybase Central 中，所有发布都显示在左窗格的 [**发布**] 文件夹中。当选中某发布时，您为该发布创建的所有项目都会出现在右窗格的 [**项目**] 选项卡中。

创建发布

基于统一数据库中的现有表创建发布。

使用以下过程创建由表中所有列和行组成的发布。

- **发布表 (Sybase Central)**

1. 以具有 DBA 权限的用户身份连接到统一数据库。
2. 在左窗格中，选择 [**发布**] 文件夹。
3. 选择 [**文件**] » [**新建**] » [**发布**]。
4. 在 [**您要给新发布指定什么名称**] 字段中键入发布的名称。单击 [**下一步**]。
5. 单击 [**下一步**]。
6. 在 [**可用表**] 列表中，选择一个表。单击 [**添加**]。

7. 单击 [**完成**]。

♦ 发布表 (SQL)

1. 以具有 DBA 权限的用户身份连接到统一数据库。
2. 执行一个 CREATE PUBLICATION 语句，该语句指定新发布的名称和要发布的表。请参见“CREATE PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。

   例如，以下语句将创建一个发布整个 Customers 表的发布：

   ```
   CREATE PUBLICATION PubCustomers (
      TABLE Customers
   );
   ```

   以下语句将创建一个发布整个 SalesOrders 表、SalesOrderItems 表和 Products 表的发布：

   ```
   CREATE PUBLICATION PubSales (
      TABLE SalesOrders,
      TABLE SalesOrderItems,
      TABLE Products
   );
   ```

只发布表中的某些列

使用以下过程创建一个包含表中所有行但仅包含某些列的发布。

♦ 只发布表中的某些列 (Sybase Central)

1. 以具有 DBA 权限的用户身份连接到统一数据库。
2. 在左窗格中，展开 [**发布**] 文件夹。
3. 从 [**文件**] 菜单选择 [**新建**] » [**发布**]。
4. 在 [**您要给新发布指定什么名称**] 字段中键入发布的名称。单击 [**下一步**]。
5. 单击 [**下一步**]。
6. 在 [**可用表**] 列表中，选择一个表。单击 [**添加**]。单击 [**下一步**]。
7. 在 [**可用列**] 选项卡上，双击表的图标以展开 [**可用列**] 列表。选择想要发布的每个列并单击 [**添加**]。单击 [**下一步**]。
8. 单击 [**完成**]。

♦ 只发布表中的某些列 (SQL)

1. 以具有 DBA 权限的用户身份连接到统一数据库。
2. 执行指定了发布名和表名的 CREATE PUBLICATION 语句。在表名后面的括号中列出发布的列。

例如，以下语句将创建一个发布 Customers 表的 ID、CompanyName 和 City 列对应的所有行的发布。该发布*不*发布 Customers 表的 Surname、GivenName、Street、State、Country、PostalCode 和 Phone 列。

```
CREATE PUBLICATION PubCustomers (
   TABLE Customers (
      ID,
      CompanyName,
      City )
);
```

另请参见

- “CREATE PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “参照完整性错误”一节第 51 页

只发布表中的某些行

要创建一个只包含表中某些行的发布，必须编写一个仅与要发布的行匹配的搜索条件。请在搜索条件中使用以下子句之一：

- **SUBSCRIBE BY 子句** 当发布的多个预订者接收表中的不同行时，则使用 SUBSCRIBE BY 子句。

  建议在 SQL Remote 系统需要大量预订时使用 SUBSCRIBE BY 子句。SUBSCRIBE BY 子句允许将多个预订与单个发布关联，而 WHERE 子句则不能。预订者将根据所提供表达式的值来接收行。

  使用 SUBSCRIBE BY 子句的发布更加紧凑，更易于理解，并且与维护若干个 WHERE 子句发布相比，具有更优越的性能。

  请参见“使用 SUBSCRIBE BY 子句只发布某些行”一节第 16 页。

- **WHERE 子句** 使用 WHERE 子句在某个项目中加入行子集。包含此项目的发布的所有预订者都将收到满足 WHERE 子句的行。

  所有未发布的行都必须具有缺省值。否则，当远程数据库尝试从统一数据库中插入新行时，将发生错误。

  可将 WHERE 子句合并到项目中。

  数据库服务器必须将信息添加到事务日志，然后扫描事务日志以便发送消息，发送的消息数与发布的数目成正比。WHERE 子句不允许将多个预订与单个发布关联，而 SUBSCRIBE BY 子句则可以。

  请参见“使用 WHERE 子句只发布某些行”一节第 17 页。

示例

您需要一个使每个销售代表执行以下任务的发布：

- 预订他们的销售订单。

- 在本地更新他们的销售订单。
- 将他们的销售数据复制到统一数据库。

如果使用 WHERE 子句，则需要为每个销售代表创建单独的发布。下面的发布是为名为 Sam Singer 的销售代表创建的；其他每个销售代表需要的发布与之类似。

```
CREATE PUBLICATION PubOrdersSamSinger (
   TABLE SalesOrders
      WHERE Active = 1
);
```

以下语句将使 Sam Singer 预订 PubsOrdersSamSinger 发布。

```
CREATE SUBSCRIPTION
TO PubOrdersSamSinger
FOR Sam_Singer;
```

如果使用 SUBSCRIBE BY 子句，则只需要创建一个发布。所有销售代表都可以使用下面的发布：

```
CREATE PUBLICATION PubOrders (
   TABLE SalesOrders
   SUBSCRIBE BY SalesRepresentativeID
);
```

以下语句将使 Sam Singer 以其 ID 号 8887 预订 PubsOrders 发布。

```
CREATE SUBSCRIPTION
TO PubOrders ('8887')
FOR Sam_Singer;
```

使用 SUBSCRIBE BY 子句只发布某些行

使用 SUBSCRIBE BY 子句通过以下过程创建一个发布。有关使用 SUBSCRIBE BY 子句及其替代子句 WHERE 子句的信息，请参见“只发布表中的某些行”一节第 15 页。

♦ 使用 SUBSRIBE BY 子句创建发布 (Sybase Central)

1. 以具有 DBA 权限的用户身份连接到统一数据库。
2. 在左窗格中，选择 [**发布**] 文件夹。
3. 从 [**文件**] 菜单选择 [**新建**] » [**发布**]。
4. 在 [**您要给新发布指定什么名称**] 字段中键入发布的名称。单击 [**下一步**]。
5. 单击 [**下一步**]。
6. 在 [**可用表**] 列表中，选择一个表。单击 [**添加**]。单击 [**下一步**]。
7. 在 [**可用列**] 选项卡上，双击表的图标以展开 [**可用列**] 列表。选择想要发布的每个列并单击 [**添加**]。单击 [**下一步**]。
8. 单击 [**下一步**]。
9. 在 [**指定 SUBSCRIBE BY 限制**] 页面上：

a. 单击 [项目] 列表中的某个表。
   单击 [列] 并从下拉列表中选择一个列。

10. 单击 [完成]。

♦ 使用 SUBSRIBE BY 子句创建发布 (SQL)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 执行包含 SUBSCRIBE BY 子句的 CREATE PUBLICATION 语句。

例如

以下语句将创建一个发布 Customers 表的 ID、CompanyName、City、State 和 Country 列的发布，并使用 State 列的值将各行与预订者进行匹配：

```
CREATE PUBLICATION PubCustomers (
   TABLE Customers (
      ID,
      CompanyName,
      City,
      State,
      Country )
   SUBSCRIBE BY State
);
```

以下语句将使两个雇员预订该发布。Ann Taylor 接收乔治亚州 (GA) 的客户，而 Sam Singer 接收麻萨诸塞州 (MA) 的客户。

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

CREATE SUBSCRIPTION
  TO PubCustomers ( 'MA' )
  FOR Sam_Singer;
```

用户可以预订多个发布，也可以具有单个发布的多个预订。

另请参见

- “使用 WHERE 子句只发布某些行”一节第 17 页
- “使用无交集数据分区”一节第 60 页
- “CREATE PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “预订发布”一节第 33 页

使用 WHERE 子句只发布某些行

使用以下过程创建一个使用 WHERE 子句以包含表中所有列但仅包含某些行的发布。有关使用 WHERE 子句及其替代子句 SUBSCRIBE BY 子句的信息，请参见“只发布表中的某些行”一节第 15 页。

♦ 使用 WHERE 子句创建发布 (Sybase Central)

1. 以具有 DBA 权限的用户身份连接到统一数据库。
2. 在左窗格中，选择 [**发布**] 文件夹。
3. 从 [**文件**] 菜单选择 [**新建**] » [**发布**]。
4. 在 [**您要给新发布指定什么名称**] 字段中键入发布的名称。单击 [**下一步**]。
5. 单击 [**下一步**]。
6. 在 [**可用表**] 列表中，选择一个表。单击 [**添加**]。单击 [**下一步**]。
7. 在 [**可用列**] 选项卡上，双击表的图标以展开 [**可用列**] 列表。选择想要发布的每个列并单击 [**添加**]。单击 [**下一步**]。
8. 在 [**指定 WHERE 子句**] 页面上：
   a. 单击 [**项目**] 列表中的某个表。
   b. 将 WHERE 子句键入 [**所选项目具有以下 WHERE 子句**] 字段。
9. 单击 [**完成**]。

♦ 使用 WHERE 子句创建发布 (SQL)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 执行使用 WHERE 子句的 CREATE PUBLICATION 语句，以包括想要包含在发布中的行。

   例如，以下语句将为在 Status 列中标记为 active 的客户创建一个发布 Customers 表的 ID、CompanyName、City、State 和 Country 列的发布。Status 列不会发布。

```
CREATE PUBLICATION PubCustomers (
   TABLE Customers (
      ID,
      CompanyName,
      City,
      State,
      Country )
   WHERE Status = 'active'
);
```

   以下语句将使两个雇员预订同一发布。Ann Taylor 和 Sam Singer 会收到相同的数据。

```
CREATE SUBSCRIPTION
TO PubCustomers
FOR Ann_Taylor;

CREATE SUBSCRIPTION
TO PubCustomers)
FOR Sam_Singer;
```

用户可以预订多个发布，也可以具有单个发布的多个预订。

请参见“CREATE PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。

另请参见

- “WHERE 子句和主键”一节第 50 页

变更发布

可通过添加、修改或删除项目或者通过重命名发布来对发布进行变更。

> **小心**
> 在运行中的 SQL Remote 系统中变更发布可能会导致复制错误和复制系统中的数据丢失。请参见“升级和重新同步”一节第 129 页。

♦ 变更发布 (Sybase Central)

1. 以拥有该发布的用户身份或具有 DBA 权限的用户身份连接到数据库。
2. 在左窗格中，选择 [**发布**] 文件夹。
3. 右击想要变更的项目并选择 [**属性**] 以编辑发布。

♦ 变更发布 (SQL)

1. 以拥有该发布的用户身份或具有 DBA 权限的用户身份连接到数据库。
2. 执行 ALTER PUBLICATION 语句。

   例如，以下语句会将 Customers 表添加到 PubContacts 发布中。

   ```
   ALTER PUBLICATION PubContacts (
      ADD TABLE Customers
   );
   ```

   例如，以下语句会在 PubContacts 发布中重新定义 Customers 项目。

   ```
   ALTER PUBLICATION PubContacts (
      ALTER ARTICLE Customers ( name, surname )
   );
   ```

另请参见

- “ALTER PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》

删除发布

删除某个发布后，该发布的所有预订也将被自动删除。

小心
在运行中的 SQL Remote 系统中删除发布可能会导致复制错误和复制系统中的数据丢失。请参见“升级和重新同步”一节第 129 页。

♦ 删除发布 (Sybase Central)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 在左窗格中，展开 [**发布**] 文件夹。
3. 右击目标发布，然后选择 [**删除**]。

♦ 删除发布 (SQL)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 执行 DROP PUBLICATION 语句。

   例如，以下语句将删除名为 PubOrders 的发布。

   ```
   DROP PUBLICATION PubOrders;
   ```

   请参见“DROP PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。

用户权限

SQL Remote 使用一致的系统来管理对远程数据库和统一数据库具有权限的用户。

SQL Remote 复制中所涉及数据库的用户通过以下其中一种或多种权限标识：

- **PUBLISH** SQL Remote 系统中的每个数据库都发布信息。因此，每个数据库都必须有发布者。要创建发布者，必须授予一个用户 PUBLISH 权限。在整个 SQL Remote 系统中，发布者用户必须是唯一的。发送数据时，发布者代表数据库。例如，当某个数据库发送消息时，该数据库的发布者用户名会包含在消息中。当某数据库收到消息时，它可以通过消息中的发布者名称来识别发送消息的数据库。
- **REMOTE** 要将消息发送给其它数据库的数据库（如统一数据库）必须指定要将消息发送给哪些远程数据库。要在统一数据库上指定这些远程数据库，必须将 REMOTE 权限授予给远程数据库的发布者。REMOTE 权限标识那些从当前数据库中接收消息的数据库。
- **CONSOLIDATE** 每个远程数据库必须指定它接收的消息所来自的统一数据库。要在远程数据库上指定统一数据库，则要将 CONSOLIDATE 权限授予给统一数据库的发布者。远程数据库只能从一个统一数据库接收消息。CONSOLIDATE 权限标识将消息发送给当前数据库的数据库。

有关这些权限的信息存储在 SQL Remote 系统表中，并且这些权限独立于其它数据库特权和权限。

抽取实用程序 (dbxtract) 自动设置权限

缺省情况下，抽取实用程序 (dbxtract) 和 [**抽取数据库向导**] 会将相应的 PUBLISH 和 CONSOLIDATE 权限授予给远程数据库中的用户。

单层层次

在单层层次中，有一个统一数据库和一个或多个位于下层的远程数据库。在这样的层次中，统一数据库将 REMOTE 权限授予给远程数据库的发布者。而每个远程数据库则将 CONSOLIDATE 权限授予给统一数据库。

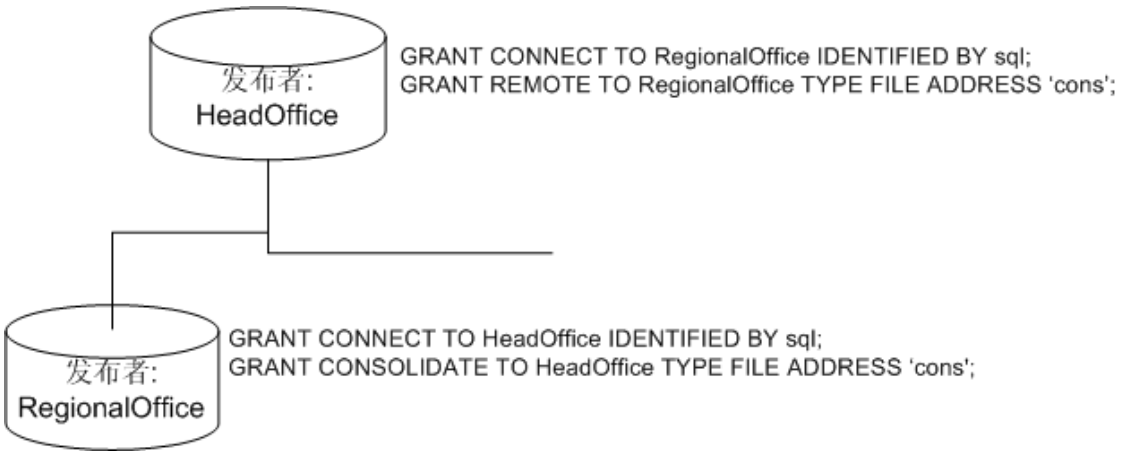
例如，有一个由其发布者 HeadOffice 标识的统一数据库和一个由其发布者 RegionalOffice 标识的远程数据库。

在统一数据库 HeadOffice 上，您可以：

- 创建一个与远程数据库的发布者同名的用户：RegionalOffice。
- 将 REMOTE 权限授予给 RegionalOffice。这会将 RegionalOffice 标识为 HeadOffice 的远程数据库。

在远程数据库 RegionalOffice 上，您可以：

- 创建一个与统一数据库的发布者同名的用户：HeadOffice。
- 将 CONSOLIDATE 权限授予给 RegionalOffice。这会将 RegionalOffice 标识为 HeadOffice 的远程数据库。



Dbxtract 自动设置权限

缺省情况下，抽取实用程序 (dbxtract) 和 [**抽取数据库向导**] 会将相应的 PUBLISH 和 CONSOLIDATE 权限授予给远程数据库中的用户。

多层层次

在多层层次中，所有直接位于当前数据库下一层的远程数据库都将被授予 REMOTE 权限。而层次中直接位于当前数据库上一层的数据库将被授予 CONSOLIDATE 权限。

例如，有一个由其发布者 HeadOffice 标识的统一数据库，该数据库有一个远程数据库 RegionalOffice。但 RegionalOffice 数据库也有一个远程数据库：Office。

在统一数据库 HeadOffice 上，您可以：

- 创建一个与远程数据库 RegionalOffice 的发布者同名的用户。
- 将 REMOTE 权限授予给用户 RegionalOffice。这会将 RegionalOffice 标识为从 HeadOffice 接收消息的数据库。

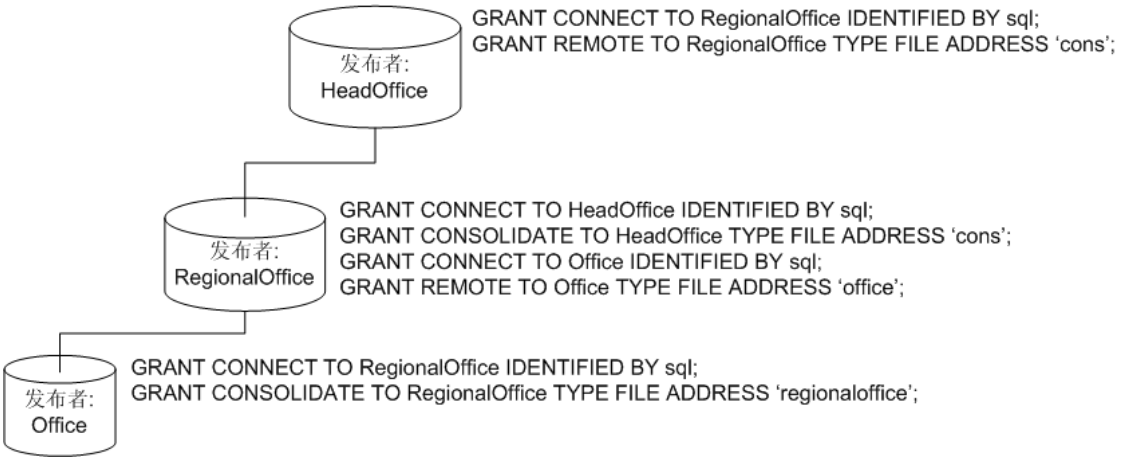
在 RegionalOffice 数据库上，您可以：

- 创建一个与统一数据库 HeadOffice 的发布者同名的用户。
- 将 CONSOLIDATE 权限授予给 HeadOffice。这会将 HeadOffice 标识为 RegionalOffice 的统一数据库；即，HeadOffice 是向 RegionalOffice 发送消息的数据库。
- 创建一个与直接位于 RegionalOffice 下层的数据库同名的用户：Office。
- 将 REMOTE 权限授予给 Office。这会将 Office 标识为从 RegionalOffice 接收消息的数据库。

在 Office 数据库上，您可以：

- 创建一个与统一数据库的发布者同名的用户：RegionalOffice。

- 将 CONSOLIDATE 权限授予给用户 RegionalOffice。这会将 RegionalOffice 标识为 Office 的统一数据库；即，RegionalOffice 向 Office 发送消息。



PUBLISH 权限

SQL Remote 系统中的每个数据库都需要发布者，发布者是具有 PUBLISH 权限的唯一用户。所有外发的 SQL Remote 消息（包括发布更新和接收确认）均由其发布者标识。SQL Remote 系统中的每个数据库都要发送接收确认。

除了在外发消息中标识发布者之外，PUBLISH 权限不具有其它任何特权。

将 PUBLISH 权限授予给具有 GROUP 权限的用户名后，该组的成员并不会继承 PUBLISH 权限。

可通过授予用户 PUBLISH 权限来创建发布者。

创建发布者

使用以下过程创建用户并授予他们 PUBLISH 权限。

♦ 创建发布者 (Sybase Central)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 创建一个用户以作为发布者（如果发布者不存在）。
   a. 在左窗格中，选择 [**用户和组**] 文件夹。
   b. 从 [**文件**] 菜单选择 [**新建**] » [**用户**]。
   c. 请按照 [**创建用户向导**] 中的说明进行操作。为用户指派口令。
3. 在 [**用户和组**] 文件夹中，右击您所创建的用户，然后选择 [**更改为发布者**]。

♦ 创建发布者 (SQL)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 执行 CREATE USER 语句以创建一个用户作为发布者。为用户指派口令。

   例如，以下语句将创建一个名为 cons、口令为 sql 的用户：

   ```
   CREATE USER cons IDENTIFIED BY SQL;
   ```
3. 执行 GRANT CONNECT 语句，将 CONNECT 权限授予给该用户。请参见“GRANT 语句”一节 《SQL Anywhere 服务器 - SQL 参考》。

   例如，以下语句将 CONNECT 权限授予给用户 cons：

   ```
   GRANT CONNECT TO cons IDENTIFIED BY SQL;
   ```
4. 执行 GRANT PUBLISH 语句，将 PUBLISH 权限授予给该用户。请参见“GRANT PUBLISH 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。

   例如，以下语句将 PUBLISH 权限授予给用户 cons：

   ```
   GRANT PUBLISH TO cons;
   ```

抽取实用程序 (dbxtract)

当某个远程数据库被抽取实用程序 (dbxtract) 或 [**抽取数据库向导**] 抽取时，远程用户会成为该远程数据库的发布者并被授予 PUBLISH 权限。

另请参见

- “撤消 PUBLISH 权限”一节第 24 页
- “查看发布者”一节第 25 页

撤消 PUBLISH 权限

使用以下过程撤消用户的 PUBLISH 权限。

> **小心**
> 如果在远程数据库或统一数据库更改发布者，可能会导致所有涉及该数据库的预订出现严重问题（包括丢失信息）。请参见“在正在运行的系统上应避免进行的更改”一节第 129 页。

♦ 撤消 PUBLISH 权限 (Sybase Central)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 在左窗格中，选择 [**用户和组**] 文件夹。
3. 右击具有 PUBLISH 权限的用户，然后选择 [**撤消发布者**]。

♦ 撤消 PUBLISH 权限 (SQL)

1. 以具有 DBA 权限的用户身份连接到数据库。

2. 执行 REVOKE PUBLISH 语句以撤消当前发布者的 PUBLISH 权限。请参见“REVOKE PUBLISH 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。

   例如：

   ```
   REVOKE PUBLISH FROM cons;
   ```

另请参见

- “PUBLISH 权限”一节第 23 页
- “查看发布者”一节第 25 页

查看发布者

♦ **标识发布者 (Sybase Central)**

- 在左窗格中，选择 [**用户和组**] 文件夹。

  在右窗格中，发布者是 [**类型**] 为 [**发布者**] 的用户。

♦ **标识发布者 (SQL)**

- 使用 CURRENT PUBLISHER 常量。以下语句检索发布者用户名：

  ```
  SELECT CURRENT PUBLISHER;
  ```

另请参见

- “PUBLISH 权限”一节第 23 页
- “撤消 PUBLISH 权限”一节第 24 页

REMOTE 权限

授予 REMOTE 权限又叫作在数据库中添加远程用户。SQL Remote 层次中直接位于当前数据库下层的数据库的发布者会被当前数据库授予 REMOTE 权限。

为用户授予 REMOTE 权限时，必须配置以下设置：

- **消息系统** 只有在数据库中至少定义了一个消息系统后，才能创建新的远程用户。请参见“SQL Remote 消息系统”一节第 105 页。
- **发送频率** 使用 SQL 语句授予 REMOTE 权限时，设置发送频率是可选操作。请参见“设置发送频率”一节第 88 页。

将 REMOTE 权限授予给用户：

- 将用户标识为远程用户。
- 指定与该远程用户交换消息时使用的消息类型。
- 提供一个消息发送的目标地址。

- 指定向远程用户发送消息的频率。

数据库的发布者不能对同一数据库同时拥有 REMOTE 和 CONSOLIDATE 权限。因为这会将发布者既标识为外发消息的发送者，又标识为这些消息的接收者。

将 REMOTE 权限授予给组

尽管可以向组授予 REMOTE 权限，但 REMOTE 权限*不*会自动应用于组中的所有用户。必须将 REMOTE 权限明确授予给组中的每个用户。

授予 REMOTE 权限

使用以下过程添加一个远程用户。

♦ 创建远程用户 (Sybase Central)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 在左窗格中，选择 [**SQL Remote 用户**] 文件夹。
3. 从 [**文件**] 菜单选择 [**新建**] » [**SQL Remote 用户**]。
4. 请按照 [**创建远程用户向导**] 中的说明进行操作。

♦ 将现有用户设置为远程用户 (Sybase Central)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 在左窗格中，选择 [**SQL Remote 用户**] 文件夹。
3. 右击该用户，然后选择 [**更改为远程用户**]。
4. 在窗口中选择消息类型、输入地址、选择发送频率并单击 [**确定**]。

♦ 创建远程用户 (SQL)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 执行 CREATE USER 语句创建一个用户。

   例如：

   ```
   CREATE USER remote1;
   ```

3. 执行 GRANT CONNECT 语句，将 CONNECT 权限授予给该用户。请参见“GRANT 语句”一节 《SQL Anywhere 服务器 - SQL 参考》 。

   例如：

   ```
   GRANT CONNECT TO remote1;
   ```

4. 执行 GRANT REMOTE 语句，将 REMOTE 权限授予给该用户。请参见“GRANT REMOTE 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》 。

   例如：


```
GRANT REMOTE TO userid;
```

例如，以下语句将 REMOTE 权限授予给用户 S_Beaulieu，并指定远程数据库执行以下任务：

- 将 SMTP 电子邮件用作其消息系统。
- 使用电子邮件地址 s_beaulieu@acme.com 发送消息：
- 每天在 10 P.M. 发送消息

```
GRANT REMOTE TO S_Beaulieu
TYPE smtp
ADDRESS 's_beaulieu@acme.com'
SEND AT '22:00';
```

以下语句将 REMOTE 权限授予给用户 rem1，并指定 rem1 的远程数据库使用地址为 rem1 的 FILE 消息系统。

```
GRANT CONNECT TO rem1 IDENTIFIED BY SQL
GRANT REMOTE TO rem1 TYPE FILE ADDRESS 'rem1';
```

另请参见

- “撤消 REMOTE 权限”一节第 27 页

撤消 REMOTE 权限

撤消用户的 REMOTE 权限：

- 从 SQL Remote 系统中删除用户。
- 将该用户或组恢复为普通用户/组。
- 取消该用户或组对所有发布的预订。

♦ 撤消 REMOTE 权限 (Sybase Central)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 在左窗格中，展开 [**用户和组**] 文件夹或 [**SQL Remote 用户**] 文件夹。
3. 右击该远程用户或组，然后选择 [**撤消远程**]。

♦ 撤消 REMOTE 权限 (SQL)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 执行 REVOKE REMOTE 语句以撤消当前用户或组的 REMOTE 权限。请参见“REVOKE REMOTE 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。

   例如，以下语句撤消用户 S_Beaulieu 的 REMOTE 权限。

   ```
   REVOKE REMOTE FROM S_Beaulieu;
   ```

另请参见

- “授予 REMOTE 权限”一节第 26 页

CONSOLIDATE 权限

SQL Remote 层次中直接位于当前数据库上层的数据库会被当前数据库授予 CONSOLIDATE 权限。在每个远程数据库中，必须为统一数据库授予 CONSOLIDATE 权限。

即使是从只读远程数据库发送到统一数据库，也必须授予 CONSOLIDATE 权限，因为接收确认是从远程数据库发送到统一数据库。

为用户授予 CONSOLIDATE 权限时，必须配置以下设置：

- **消息系统** 只有在数据库中至少定义了一个消息系统后，才能创建新的统一用户。请参见“SQL Remote 消息系统”一节第 105 页。
- **发送频率** 使用 SQL 语句授予 CONSOLIDATE 权限时，设置发送频率是可选操作。请参见“设置发送频率”一节第 88 页。

授予 CONSOLIDATE 权限：

- 将用户标识为统一用户。
- 指定与该统一用户交换消息时使用的消息类型。
- 提供一个消息发送的目标地址。
- 指定向统一用户发送消息的频率。

数据库的发布者不能对同一数据库同时拥有 REMOTE 和 CONSOLIDATE 权限。因为这会将发布者既标识为外发消息的发送者，又标识为这些消息的接收者。

抽取实用程序 (dbxtract)

当某个远程数据库被抽取实用程序 (dbxtract) 或 [**抽取数据库向导**] 抽取时，会自动对远程数据库执行 GRANT CONSOLIDATE 语句。

授予 CONSOLIDATE 权限

使用以下过程将 CONSOLIDATE 权限授予给用户。

建议将 CONSOLIDATE 权限授予给统一数据库的发布者。

♦ 指定统一数据库 (Sybase Central)

1. 以具有 DBA 权限的用户身份进行连接。
2. 在左窗格中，单击数据库，然后选择 [**文件**] » [**属性**]。
3. 单击 [**SQL Remote**] 选项卡。

4. 选择 [**此远程数据库具有对应的统一数据库**]。
5. 配置 [**消息类型**]、[**地址**] 和 [**发送频率**] 设置。
6. 单击 [**确定**] 以关闭 [**统一数据库属性**] 窗口。

♦ 指定统一数据库 (SQL)

1. 执行 GRANT CONNECT 语句，将 CONNECT 权限授予给统一数据库的发布者。请参见“GRANT 语句”一节 《SQL Anywhere 服务器 - SQL 参考》。

   例如：

   ```
   GRANT CONNECT TO cons;
   ```

2. 执行 GRANT CONSOLDIATE 语句，将 CONSOLIDATE 权限授予给该统一用户。请参见“GRANT CONSOLIDATE 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。

   例如，以下语句会将 CONSOLIDATE 权限授予给用户 hq_user，并指定统一数据库使用 SMTP 电子邮件系统：

   ```
   GRANT CONSOLIDATE TO hq_user
   TYPE SMTP
   ADDRESS 'hq_address';
   ```

   该 GRANT CONSOLIDATE 语句中没有 SEND 子句，因此，SQL Remote 会将消息发送到统一数据库，然后停止。

   例如，以下语句会将 CONSOLIDATE 权限授予给用户 cons，并指定统一数据库使用 FILE 系统：

   ```
   GRANT CONNECT TO "cons" IDENTIFIED BY SQL;
   GRANT CONSOLIDATE TO "cons" TYPE "FILE" ADDRESS 'cons';
   GRANT CONNECT TO "rem1" IDENTIFIED BY SQL;
   GRANT PUBLISH TO "rem1";
   CREATE REMOTE MESSAGE TYPE FILE ADDRESS 'rem1';
   ```

另请参见

- “撤消 CONSOLIDATE 权限”一节第 29 页

撤消 CONSOLIDATE 权限

在撤消用户的 CONSOLIDATE 权限时，SQL Anywhere 会执行以下任务：

- 将用户从 SQL Remote 系统中删除。
- 将该用户或组恢复为普通用户/组。
- 取消该用户或组对所有发布的预订。

♦ 撤消 CONSOLDIATE 权限 (Sybase Central)

1. 以具有 DBA 权限的用户身份连接到数据库。

2. 展开 [**用户和组**] 文件夹或 [**SQL Remote 用户**] 文件夹。
3. 右击该统一用户或组，然后选择 [**撤消统一**]。
4. 单击 [**是**]。

♦ 撤消 CONSOLIDATE 权限 (SQL)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 执行 REVOKE CONSOLIDATE 语句以撤消当前用户或组的 CONSOLIDATE 权限。请参见“REVOKE CONSOLIDATE 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。

   例如：

   ```
   REVOKE CONSOLIDATE FROM Cons_User;
   ```

另请参见

- “授予 CONSOLIDATE 权限”一节第 28 页

授予 REMOTE DBA 权限

具有 REMOTE DBA 权限的用户名只有从 SQL Remote 连接时，才对数据库拥有完全的 DBA 权限。通过 REMOTE DBA 权限，SQL Remote 拥有对数据库的完全访问权，并且可以在消息中进行指定的更改。只有具有 REMOTE DBA 或 DBA 权限的用户才能运行 SQL Remote。

作为一个特殊权限，REMOTE DBA 具有以下属性：

- **不通过 SQL Remote 连接时没有确切的权限** 被授予 REMOTE DBA 权限的用户对与 SQL Remote 分离的连接没有额外的特权。因此，即使 REMOTE DBA 用户的用户名和口令被广泛分发，也不存在安全问题。只要用户名没有被授予除了针对数据库授予的 CONNECT 权限之外的权限，就没有人可以使用该用户名访问数据库中的数据。
- **SQL Remote 的完全 DBA 权限** 在从 SQL Remote 连接时，具有 REMOTE DBA 权限的用户对该数据库拥有完全 DBA 权限。

何时授予 REMOTE DBA 权限

建议在统一数据库上创建用户时将 REMOTE DBA 权限授予给统一数据库的发布者以及每个远程用户。当某个远程数据库被抽取实用程序 (dbxtract) 或 [**抽取数据库向导**] 抽取时，远程用户会成为该远程数据库的发布者并被授予 PUBLISH 权限和 REMOTE DBA 权限。

此推荐做法将简化用户的管理。每个远程用户只需要一个用户名即可连接到数据库，无论是从 SQL Remote 连接（这使用户具有完全的 DBA 权限），还是从其它任何客户端应用程序连接（这种情况下，REMOTE DBA 权限不向用户授予任何额外权限）。

♦ 授予 REMOTE DBA 权限 (Sybase Central)

1. 以具有 DBA 权限的用户身份连接到数据库。

2. 在左窗格中，选择 [**用户和组**] 文件夹或 [**SQL Remote 用户**] 文件夹。
3. 右击用户并选择 [**属性**]。
4. 单击 [**授权**] 选项卡并选择 [**远程 DBA**] 选项。
5. 单击 [**应用**]，然后单击 [**确定**]。

♦ 授予 REMOTE DBA 权限 (SQL)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 执行 GRANT REMOTE DBA 语句，将 REMOTE DBA 权限授予给用户。

   例如：

```
GRANT REMOTE DBA TO dbremote
IDENTIFIED BY dbremote;
```

另请参见

- “REMOTE DBA 特权”一节 《SQL Anywhere 服务器 - 数据库管理》
- “GRANT REMOTE DBA 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》

撤消 REMOTE DBA 权限

从 SQL Remote 连接时，撤消 REMOTE DBA 权限会使用户无法对数据库拥有完全 DBA 权限。请参见“授予 REMOTE DBA 权限”一节第 30 页。

使用以下过程撤消用户的 REMOTE DBA 权限。

♦ 撤消 REMOTE DBA 权限 (Sybase Central)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 在左窗格中，选择 [**用户和组**] 文件夹或 [**SQL Remote 用户**] 文件夹。
3. 右击用户并选择 [**属性**]。
4. 单击 [**授权**] 选项卡并清除 [**远程 DBA**] 选项。
5. 单击 [**确定**]。

♦ 撤消 REMOTE DBA 权限 (SQL)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 执行 REVOKE REMOTE DBA 语句以撤消用户的 REMOTE DBA 权限。

   例如：

```
REVOKE REMOTE DBA FROM dbremote;
```


另请参见

- “REVOKE REMOTE DBA 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “授予 REMOTE 权限”一节第 26 页

预订

可通过创建**预订**为用户预订一个发布。在发布中共享信息的每个数据库都必须预订该发布。

对数据库中每个发布所做的更改会定期复制给该发布的所有预订者。这些复制称为**发布更新**。

预订发布

要为某用户预订一个发布，您需要以下信息：

- **用户名** 预订发布的用户。此用户必须已被授予 REMOTE 权限。
- **发布名称** 用户所预订的发布的名称。
- **预订值** 只有在发布中包含 SUBSCRIBE BY 子句时，预订值才适用。预订值是对照发布的 SUBSCRIBE BY 子句进行测试的值。例如，如果某发布将包含雇员 ID 的列的名称作为 SUBCRIBE BY 子句，则在创建预订时必须提供预订用户的雇员 ID 的值。预订值始终是一个字符串。

  只有在发布中含有 SUBSCRIBE BY 子句时才需要此值。请参见“使用 SUBSCRIBE BY 子句只发布某些行”一节第 16 页。

♦ 为用户预订发布 (Sybase Central)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 在左窗格中，选择 [**发布**] 文件夹。
3. 选择一个发布。
4. 在右窗格中，单击 [**SQL Remote 预订**] 选项卡。
5. 从 [**文件**] 菜单选择 [**新建**] » [**SQL Remote 预订**]。
6. 请按照 [**创建 SQL Remote 预订向导**] 中的说明进行操作。

   预订的详细信息视发布是否使用预订表达式而异。

♦ 为远程用户预订发布 (SQL)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 执行 CREATE SUBSCRIPTION 语句，为用户预订一个发布。

   例如，以下语句将为用户名 SamS 创建一个对 CustomerPub 发布的预订（该发布是使用 WHERE 子句创建的）：

   ```
   CREATE SUBSCRIPTION
     TO CustomerPub
     FOR SamS;
   ```

   例如，以下语句将为用户名 SamS 创建一个对 PubOrders 发布的预订，该预订由预订表达式 SalesRepresentative 定义，请求 Sam Singer 自己的销售信息的数据行：

```
CREATE SUBSCRIPTION
  TO PubOrders ( '856' )
  FOR SamS;
```

另请参见

- “CREATE SUBSCRIPTION 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “使用 SUBSCRIBE BY 子句只发布某些行”一节第 16 页

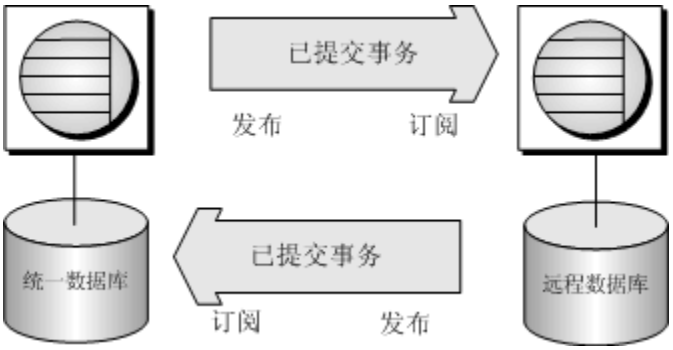
了解基于事务日志的复制

SQL Remote 会复制：

- **已提交的更改** 已对数据库进行的更改会记录在它们的事务日志中。
- **修改属于发布的数据的那些更改** SQL Remote 会扫描事务日志以找出对属于发布的数据行的已提交更改，将 SQL 语句打包成消息，并将消息发送到预订者数据库。

在统一数据库上，事务日志中所有属于发布的已提交事务都会被定期发送到远程数据库。

在远程数据库上，事务日志中所有属于发布的已提交事务都会被定期发送到统一数据库。



数据库服务器处理发布

SQL Anywhere 数据库服务器是计算发布并将信息写入事务日志的组件。发布越多，数据库服务器要做的工作就越多。

SQL Anywhere 为每次对表（属于发布的表）的更新计算预订表达式。它向事务日志中添加更新前后表达式的值。对于属于多个发布的表，在更新前后为每个发布都计算预订表达式。

事务日志中的额外信息可能在下列情况下影响性能：

- **计算费时的表达式** 如果计算预订表达式十分费时，则它可能会影响性能。
- **多个发布** 当一个表属于多个发布时，必须计算多个表达式。相反，*预订* 的数量与数据库服务器无关。
- **多值表达式** 一些表达式是多值的，这会导致事务日志中额外增加信息。这样会影响性能。

预订由 SQL Remote 处理

SQL Remote 是执行语句复制的组件。

在发送阶段，SQL Remote 消息代理将当前预订映射到事务日志中的发布信息并为每个远程用户生成相应的消息。请参见“消息代理 (dbremote)”一节第 91 页。

另请参见

- “事务日志”一节 《SQL Anywhere 服务器 - 数据库管理》

复制 INSERT 和 DELETE 语句

对于 SQL Remote，INSERT 和 DELETE 语句是要复制的最简单的语句：

- 对某一数据库执行 INSERT 语句时，该语句会作为 INSERT 语句发送到 SQL Remote 系统中的预订数据库。
- 对某一数据库执行 DELETE 语句时，该语句会作为 DELETE 语句发送到 SQL Remote 系统中的预订数据库。

统一数据库

SQL Remote 复制统一数据库事务日志中的每个 INSERT 或 DELETE 语句，并将其发送到预订了要被插入或删除的行的每个远程数据库。如果只预订了表中列的子集，则发送到远程数据库的 INSERT 语句将仅包含这些列。

远程数据库

SQL Remote 复制远程数据库事务日志中的每个 INSERT 或 DELETE 语句，并将其发送到预订了要被插入或删除的行的统一数据库。随后，统一数据库会应用该语句，这将导致写入其事务日志。在 SQL Remote 处理统一数据库事务日志时，这些更改最终会被传播到其它远程站点。SQL Remote 可以确保不会将语句发送给最初执行它们的远程用户。

另请参见

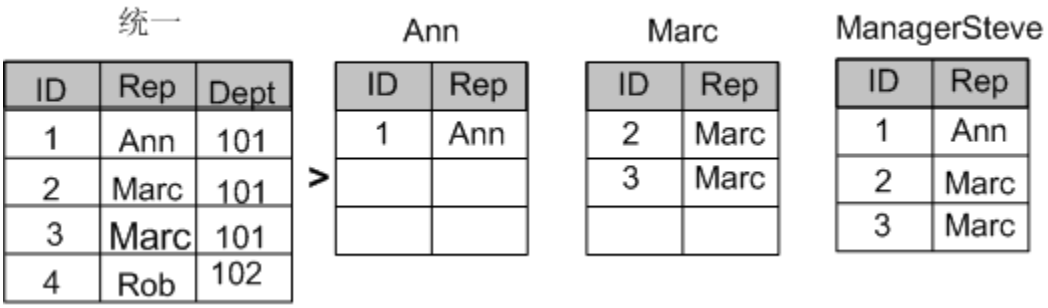
- “重复主键错误”一节第 53 页
- “未找到行错误”一节第 50 页

复制 UPDATE 语句

UPDATE 语句在被复制时可能不会完全与输入数据库时一样。以下情形介绍了 UPDATE 语句是如何被复制的：

- 如果某 UPDATE 语句具有更新给定远程用户的预订中的行的作用，则将其作为 UPDATE 语句发送给该用户。
- 如果某 UPDATE 语句具有从给定远程用户的预订中删除行的作用，则将其作为 DELETE 语句发送给该用户。
- 如果某 UPDATE 语句具有向给定远程用户的预订中添加行的作用，则将其作为 INSERT 语句发送给该用户。

为了演示如何复制 UPDATE 语句，以下示例将对用户 Ann、Marc 和 ManagerSteve 使用一个统一数据库和三个远程数据库。



在统一数据库上，有一个名为 cons 的发布，该发布是用以下语句创建的：

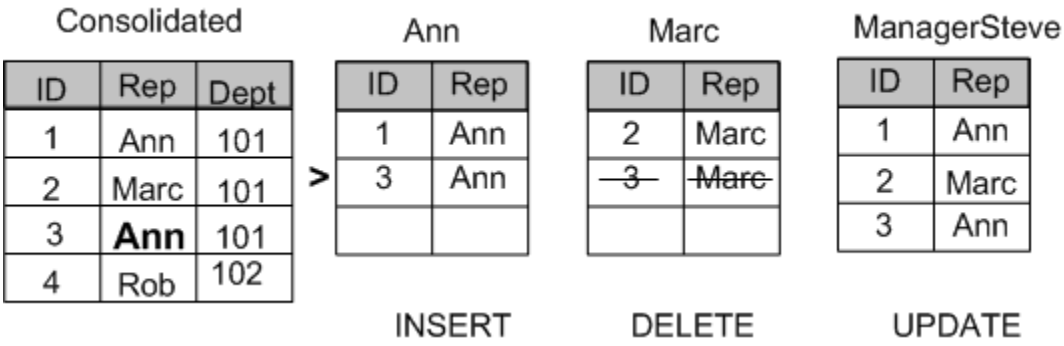
```
CREATE PUBLICATION "cons"."p1" (
 TABLE "DBA"."customers" ( "ID", "Rep") SUBSCRIBE BY repid
);
```

Ann 和 Marc 通过他们各自的 Rep 列值预订 cons 发布。而 ManagerSteve 使用 Ann 和 Marc 的 Rep 列值预订 cons 发布。以下语句将使三个用户预订发布 cons：

```
CREATE SUBSCRIPTION
  TO "cons"."p1"( 'Ann' )
  FOR "Ann";
CREATE SUBSCRIPTION
  TO "cons"."p1"( 'Marc' )
  FOR "Marc";
CREATE SUBSCRIPTION
  TO "cons"."p1"( 'Ann' )
  FOR "ManagerSteve";
CREATE SUBSCRIPTION
  TO "cons"."p1"( 'Marc' )
  FOR "ManagerSteve";
```

在统一数据库上，将某行的 Rep 值从 Marc 更改为 Ann 的 UPDATE 语句被复制到：

- Marc（作为 DELETE 语句）。
- Ann（作为 INSERT 语句）。
- ManagerSteve（作为 UPDATE 语句）。



这种在预订者之间重新指派行的做法有时称为**地域调整**，因为它是销售人员自动应用程序的常见功能，在这种功能中客户被定期重新指派给销售代表。

另请参见

- “更新冲突”一节第 43 页

复制过程

SQL Remote 通过复制过程的操作来复制过程。不复制过程调用，而是复制过程的各个操作（INSERT、UPDATE 和 DELETE 语句）。

复制触发器

通常，远程数据库与统一数据库定义了相同的触发器。

缺省情况下，SQL Remote 不复制由触发器执行的操作。实际上，当在远程数据库上对在统一数据库中触发触发器的操作进行复制时，复制触发器将在远程数据库上自动触发。这将避免权限问题以及每个操作发生两次的可能性。此规则有一些例外情况：

- **RESOLVE UPDATE 触发器的复制** 由冲突解决或 RESOLVE UPDATE 触发器执行的操作将从统一数据库*复制*到所有远程数据库，包括发送引起冲突的消息的远程数据库。请参见“更新冲突的缺省解决方法”一节第 43 页。
- **BEFORE 触发器的复制** 修改要更新的行的 BEFORE 触发器的操作会在执行 UPDATE 语句操作前进行*复制*。

  例如，增加行中计数器列以跟踪行的更新次数的 BEFORE UPDATE 触发器如果被复制，则将进行双倍计数，因为复制 UPDATE 语句时将在远程数据库上触发 BEFORE UPDATE 触发器。

  将列设置为上次更新时间的 BEFORE UPDATE 触发器也将获取复制 UPDATE 语句的时间。

  若要防止此问题，必须确保预订者数据库中没有 BEFORE UPDATE 触发器，或该触发器不执行被复制的操作。

复制触发器操作的选项

要在发送消息时复制所有触发器操作，请使用消息代理 (dbremote) 的 -t 选项。请参见“消息代理 (dbremote)”一节第 140 页。

使用 -t 选项时，请确保触发器操作不会在远程数据库中执行两次（一次在执行复制触发器操作时，一次在远程数据库上触发触发器时）。

要确保触发器操作不会执行两次，请使用以下选项之一：

- 使用 IF CURRENT REMOTE USER IS NULL ...END IF 语句包裹触发器主体。
- 将 SQL Remote 用户名的 SQL Anywhere fire_triggers 选项设置为 Off。请参见“fire_triggers 选项 [兼容性]”一节 《SQL Anywhere 服务器 - 数据库管理》。

避免触发器错误

如果发布只包含数据库的子集，则统一数据库中的触发器可能仅引用存在于统一数据库而不存在于远程数据库的表或行。当这样的触发器在远程数据库上触发时，会发生错误。要避免这些错误，请使用 IF 语句以使触发器操作附有条件，并：

- 使触发器的操作以 CURRENT PUBLISHER 的值为条件。请参见“CURRENT PUBLISHER 特殊值”一节 《SQL Anywhere 服务器 - SQL 参考》。
- 使触发器的操作以不返回 NULL 的 object_id 函数为条件。object_id 函数将表或其它对象视为参数，并返回该对象的 ID 号；如果该对象不存在，则返回 NULL。请参见“系统函数”一节 《SQL Anywhere 服务器 - SQL 参考》。
- 使触发器的操作以确定行是否存在的 SELECT 语句为条件。

抽取实用程序 (dbxtract)

缺省情况下，数据库抽取实用程序 (dbxtract) 和 [**抽取数据库向导**] 会抽取触发器定义。

另请参见

- “使用 CURRENT REMOTE USER 特殊常量”一节第 47 页

数据定义语句

除非在直通模式下执行数据定义语句（CREATE、ALTER 和 DROP），否则 SQL Remote 将不复制这些语句。

请参见“SQL Remote 直通模式”一节第 130 页。

数据类型

SQL Remote 不执行任何字符集转换。

使用兼容的排序顺序和字符集

SQL Anywhere 统一数据库使用的字符集和归类必须与远程数据库使用的相同。有关受支持的字符集的信息，请参见“国际语言和字符集” 《SQL Anywhere 服务器 - 数据库管理》。

BLOB

BLOB 包括 LONG VARCHAR、LONG BINARY、TEXT 和 IMAGE 数据类型。

当 SQL Remote 复制 INSERT 或 UPDATE 语句时，它使用变量来替代 BLOB 值。也就是说，将 BLOB 分成若干片段，并按块复制。在接收者数据库中，这些片段将进行重组（通过使用 SQL 变量）和连接。变量的值由以下形式的一系列语句构成：

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

该变量可使涉及长值的 SQL 语句长度变短，从而使它们能够容纳于单条消息内。

SET 语句是单独的 SQL 语句，因而可将 BLOB 有效地拆分成几个 SQL Remote 消息。

控制 BLOB 的复制

SQL Anywhere 的 blob_threshold 选项使您可以进一步控制长值的复制。任何长度超过 blob_threshold 选项的值都会像 BLOB 值那样进行复制。请参见“blob_threshold 选项 [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》。

使用 verify_threshold 选项最大程度地减小消息大小

Verify_threshold 数据库选项可防止验证长值（在复制的 UPDATE 语句的 VERIFY 子句中）。该选项的缺省值为 1000。如果列的数据类型比该阈值长，则当复制 UPDATE 语句时将不会验证列的旧值。这样可以减小 SQL Remote 消息的大小，但弊端是检测不到冲突的长值更新。

在将 verify_threshold 用于减小消息大小时，可使用以下方法检测冲突。

♦ 设置 verify_threshold 时检测 BLOB 值的冲突

1. 配置您的数据库，以便在更新 BLOB 时，也会更新同一表中的 last_modified 列。
2. 配置您的发布，以便 last_modified 列能与 BLOB 列一起复制。

   复制 BLOB 列和 last_modified 列时，可对 last_modified 列中的值进行验证。如果与 last_modified 列有冲突，则与 BLOB 列同样有冲突。

请参见“verify_threshold 选项 [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》。

使用工作表避免冗余更新

对 BLOB 的重复更新应在工作表中进行，并将最终版本指派给被复制的表。例如，如果正在进展中的某文档在整天内更新了 20 次并且 SQL Remote 在这一天结束时运行了一次，则将复制所有的 20 次更新。如果该文档的长度为 200 KB，则将发送 4 MB 的消息。

建议使用 *document_in_progress* 表。当用户修订完文档时，应用程序会将其从 *document_in_progress* 表移动到复制的表中。由此得到单个更新（200 KB 的消息）。

另请参见

- “SET 语句”一节 《SQL Anywhere 服务器 - SQL 参考》

日期和时间

复制日期或时间列时，SQL Remote 使用 sr_date_format、sr_time_format 和 sr_timestamp_format 数据库选项的设置来设定日期格式。

例如，以下选项设置指示 SQL Remote 将 1998 年 5 月 2 日这一日期以 1998-05-02 的形式发送。

```
SET OPTION sr_date_format = 'yyyy-mm-dd';
```

复制日期和时间时：

- 时间、日期和时间戳的格式在整个 SQL Remote 系统中必须保持一致。
- 日期和时间戳格式使用的年、月、日顺序必须与 date_order 数据库选项的设置相符。
- 可以更改每次连接持续时间的 date_order 选项。

另请参见

- “sr_date_format 选项 [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》
- “sr_time_format 选项 [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》
- “sr_timestamp_format [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》
- “date_order 选项 [数据库]”一节 《SQL Anywhere 服务器 - 数据库管理》

复制冲突和错误

SQL Remote 允许在多个数据库更新数据库。设计时须小心谨慎以避免复制错误，特别是在数据库结构复杂时更是如此。

复制冲突

复制冲突和错误不同。如果得以妥善处理，冲突在 SQL Remote 中算不上是问题。

冲突发生在许多系统中。SQL Remote 允许使用触发器和过程在 SQL Remote 系统的常规操作中对冲突进行相应的解决。请参见“更新冲突的缺省解决方法”一节第 43 页。

复制错误

复制错误分为以下几类：

- **未找到行错误** 某用户删除某行（具有给定主键值）。第二个用户在另一站点更新或删除同一行。在这种情况下，第二个语句失败，因为找不到该行。请参见“未找到行错误”一节第 50 页
- **参照完整性错误** 当包含某外键的列包括在发布中但关联的主键未包括在发布中时，引用此外键的 INSERT 语句会失败。

  当主表具有 SUBSCRIBE BY 表达式，而相关外表没有时，也可发生参照完整性错误：可能复制外表中的行，但主表中的行可能从发布中被排除。

  请参见“参照完整性错误”一节第 51 页。
- **重复主键错误** 两个用户使用相同的主键值插入某行，或者一个用户更新主键，而第二个用户插入具有新值的主键。复制系统中到达给定数据库的第二个操作失败，因为它将产生重复的主键。请参见“重复主键错误”一节第 53 页。

传送错误

有关传送错误和如何处理它们的信息，请参见“了解保证消息传送系统”一节第 99 页。

另请参见

- “报告并处理复制错误”一节第 123 页

更新冲突

如果共享数据只是为了进行读取，或每一行（由其主键标识）只在一个数据库更新，则不会发生更新冲突。只有数据在多个数据库进行更新时才会发生更新冲突。

要复制 UPDATE 语句，SQL Remote 需要为每行发出一个单独的 UPDATE 语句。这些单行语句可能会由于以下原因之一而失败：

- **要更新的行在一列或多列中不同**　当预期出现的值之一已被其他某用户更改时，会出现**更新冲突**。

  在远程数据库上，无论行中的值是什么，都将发生更新。

  在统一数据库上，SQL Remote 允许进行**冲突解决**操作。例如，在检测到冲突时，统一数据库可以：

  - 使用缺省冲突解决方法。请参见“更新冲突的缺省解决方法”一节第 43 页。
  - 使用采用 VERIFY 子句的自定义冲突解决方法。请参见“使用 VERIFY 子句的自定义冲突解决方法”一节第 44 页。
  - 使用采用触发器的自定义冲突解决方法。“使用触发器的自定义冲突解决方法”一节第 46 页.

  冲突解决不适用于主键更新
  UPDATE 语句冲突*不*适用于主键更新。您不应在 SQL Remote 系统中更新主键。必须通过适当的设计将主键冲突排除在系统之外。

- **要更新的行不存在**　每一行均由其主键值标识。如果行已被删除或主键已被另一个用户更改，则无法找到要更新的行。

  在远程数据库上，不会发生更新。

  在统一数据库上，也不会发生更新。

- **没有主键或唯一约束的表在复制更新的 WHERE 子句中引用所有列**　当两个远程数据库对同一行各自进行单独的更新并将更改复制到统一数据库时，将应用第一个到达统一数据库的更改，而不会应用来自第二个数据库的更改。

  这样，数据库将变得不一致。所有复制的表都应具有主键或唯一性约束，且约束中的列决不能更新。

更新冲突的缺省解决方法

只有数据在多个站点进行更新时才会发生更新冲突。例如：

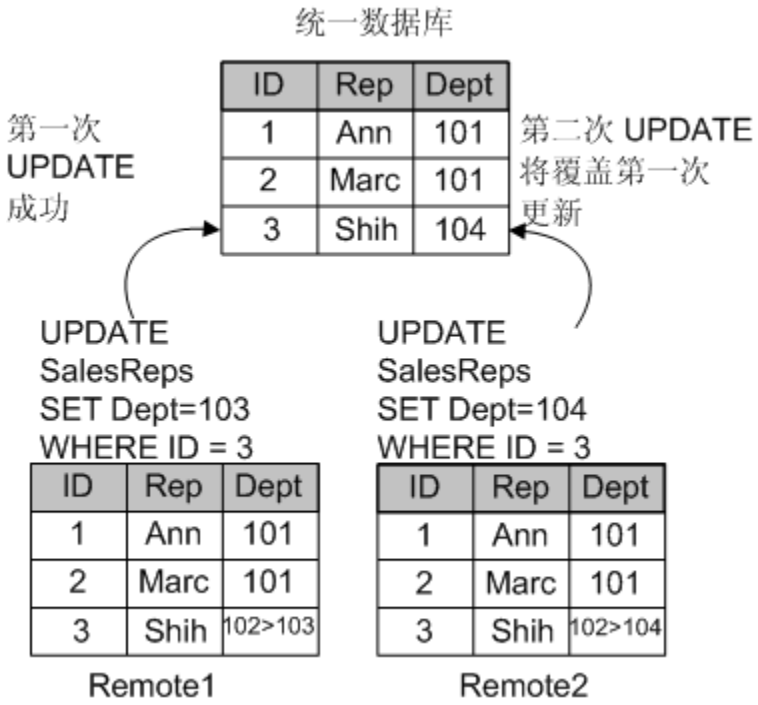
1. 用户 1 在远程站点 1 更新行。
2. 用户 2 在远程站点 2 更新相同的行。
3. 来自用户 1 的更新被发送和应用到统一数据库。
4. 来自用户 2 的更新被发送到统一数据库。

解决此类更新冲突的**缺省**方法如下：

a. 时间更近的操作（在本例中为来自用户 2 的操作）将成功执行。它会成为统一数据库中的值，并会被复制到被预订到该行的所有其它数据库中。

b. 所有其它的更新（在本例中为来自用户 1 的更新）将丢失。

c. 不会对冲突进行报告。



使用 VERIFY 子句的自定义冲突解决方法

SQL Remote 会在消息中生成使用 VERIFY 子句的 UPDATE 语句。UPDATE 语句将一行或多行从现有值更改为新值。包括 VERIFY 子句的 UPDATE 语句还包含行的现有值。

在应用 UPDATE 语句时，统一数据库会将行的现有值与远程数据库对行的现有值所期望变成的值进行比较。当 VERIFY 子句的值和数据库中的行不匹配时，数据库服务器会检测到更新冲突。

例如，下列事件发生时将出现更新冲突：

1. 用户 1 在远程站点 1 更新行。
2. 用户 2 在远程站点 2 更新相同的行。
3. 来自用户 1 的更新被发送和应用到统一数据库。
4. 来自用户 2 的更新被发送到统一数据库。

因为 UPDATE 语句包含 VERIFY 子句，所以统一数据库可检测到冲突。在统一数据库上，SQL Remote 会将它的行值与用户 2 发送的旧行值进行比较。当这些值不相同时，会存在更新冲突。

在检测到更新冲突时，统一数据库会执行以下任务：

a. 触发为该操作定义的任何冲突解决触发器。

可定义**冲突解决触发器**以处理更新冲突。当远程用户应用消息时，冲突解决触发器只在统一数据库中触发。请参见“使用触发器的自定义冲突解决方法”一节第 46 页。

b. 执行 UPDATE 语句。

c. 将冲突解决触发器的所有操作以及 UPDATE 语句发送到所有远程数据库，其中包括触发该冲突的消息的发送方。

通常，SQL Remote 不复制触发器的操作，并假定触发器存在于远程数据库上。冲突解决触发器只在统一数据库上触发，因此，这些触发器的操作会被复制到远程数据库。

5. 远程数据库从统一数据库接收 UPDATE 语句。

在远程数据库上，当来自统一数据库的消息包含更新冲突时，不会触发 RESOLVE UPDATE 触发器。

6. 在远程数据库上处理 UPDATE 语句。

在此过程的末尾，数据在整个系统中达到一致。

包含 VERIFY 子句的 UPDATE 语句

带有 VERIFY 子句的 UPDATE 语句采用以下形式：

```
UPDATE table-list
SET column-name = expression, ...
[ VERIFY (column-name, ...)
   VALUES ( expression, ...) ]
[ WHERE search-condition ]
[ ORDER BY expression [ ASC | DESC ], ... ]
```

只有在 *table-list* 参数由单个表组成时，才能使用 VERIFY 子句。它将指定列的值与一组预期值（这些值是应用 UPDATE 语句时存在于发布者数据库中的值）进行比较。当指定 VERIFY 子句时，一次只能更新一个表。

VERIFY 子句只适用于单行更新。但是，对数据库执行的多行 UPDATE 语句会被 SQL Remote 作为一组单行 UPDATE 语句来复制，因此这不会对客户端应用程序施加任何约束。

另请参见

- “UPDATE 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》

使用 VERIFY_ALL_COLUMNS 选项

缺省情况下，数据库选项 verify_all_columns 为 Off。如果将其设置为 Off，则只检查那些更新的列。当 verify_all_columns 选项被设置为 On 时：

- 对复制的 UPDATE 语句验证所有列。
- 只要有任何列不相同，就会触发 RESOLVE UPDATE 触发器。
- 消息大小会增大，因为要为每个 UPDATE 语句发送更多信息。

可以为 PUBLIC 组或只为包含在 SQL Remote 连接字符串中的用户设置 verify_all_columns 选项。请参见“verify_all_columns 选项 [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》。

抽取实用程序 (dbxtract)

当在抽取远程数据库前在统一数据库上设置 verify_all_columns 选项时，远程数据库上的 verify_all_columns 选项将由抽取实用程序 (dbxtract) 和 [**抽取数据库向导**] 进行设置。

使用触发器的自定义冲突解决方法

自定义冲突解决方法可以采取多种形式。例如，在某些应用程序中，可通过以下方法解决冲突：

- 比较原始事务的日期。请参见“解决日期冲突”一节第 47 页。
- 对两个或多个更新的结果执行计算。请参见“解决库存冲突”一节第 47 页。
- 在表中报告冲突。请参见“使用 CURRENT REMOTE USER 特殊常量”一节第 47 页。

自定义冲突解决方法需要您编写 RESOLVE UPDATE 触发器。

使用 RESOLVE UPDATE 冲突解决触发器

RESOLVE UPDATE 触发器在每行更新之前触发。RESOLVE UPDATE 触发器的语法如下所示：

```
CREATE TRIGGER trigger-name
RESOLVE UPDATE
OF column-name ON table-name
[ REFERENCING [ OLD AS old-val ]
   [ NEW AS new-val ]
     [ REMOTE AS remote-val ] ]
FOR EACH ROW
BEGIN
  ...
END
```

REFERENCING 子句允许访问表中要更新的行中的值 (OLD)、行将更新成的值 (NEW) 和根据 VERIFY 子句 (REMOTE) 应该存在的行。REMOTE AS 子句中只能引用存在于 VERIFY 子句中的列；其它列将返回一个 [**未找到列**] 错误。

使用 CURRENT REMOTE USER 特殊常量

当 CURRENT REMOTE USER 执行未提供文档的 REMOTE USER 语句时，它由 DBREMOTE 来设置。该值为 NULL，但从 DBREMOTE 进行的连接除外。

可在将冲突报告存放到某个表中的 RESOLVE UPDATE 触发器中使用 CURRENT REMOTE USER，以标识产生冲突的用户。

另请参见

- “CREATE TRIGGER 语句”一节 《SQL Anywhere 服务器 - SQL 参考》
- “UPDATE 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》

解决日期冲突

为了说明如何解决日期冲突，假设联系人管理系统中某个表的一个列保存着与每个客户的最新联系信息。

某销售代表在星期五与某客户进行了业务洽谈，但直到下个星期一才将更改上载到统一数据库。同时，另一个销售代表在星期六会见了该客户，并在当晚更新了更改。

当星期六的更新复制到统一数据库时没有冲突，但当星期一的更新到达时，它发现该行已经更改。

缺省情况下，星期一的更新将继续进行，在该列上留下不正确的信息，即，最近的联系发生在星期五。但是，应该通过在行中插入最新日期来解决此列上的更新冲突。

实现解决方案

以下 RESOLVE UPDATE 触发器选择两个新值中较新的一个并将其输入到数据库中。

```
CREATE TRIGGER contact_date RESOLVE UPDATE
 ON Contacts
 REFERENCING OLD AS old_name
  NEW AS new_name
 FOR EACH ROW
 BEGIN
   IF new_name.contact_date <
      old_name.contact_date THEN
          SET new_name.contact_date
           = old_name.contact_date
   END IF
 END;
```

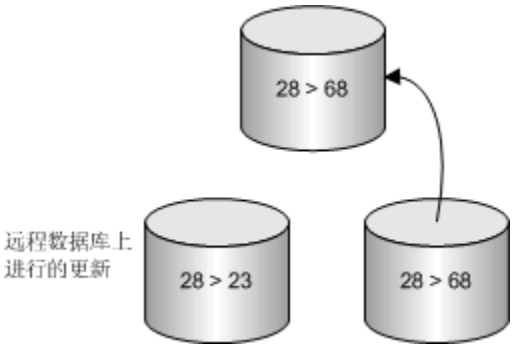
如果要更新的值时间晚于要替换它的值，则新值将被重设，使该条目保持不变。

解决库存冲突

假设有一个体育用品制造商的仓库系统。有一个产品信息表，其中的 Quantity 列保存每种产品的库存数量。对此列的更新通常是减少库存量，如果是有新货物入库，则增加库存量。

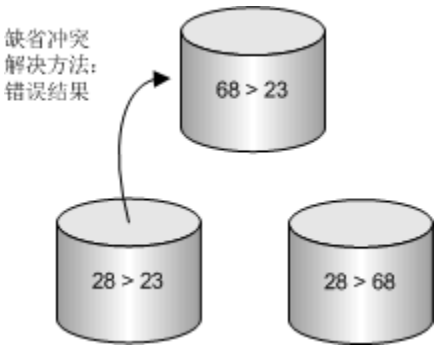
某远程数据库的销售代表输入一个订单，将小号无袖紧身 T 恤衫的库存减少 5 件（从 28 减到 23），并在她的数据库中输入此信息。同时，在将此更新复制到统一数据库之前，另一个销售代表

收到了 40 件退回的 T 恤衫。该销售代表在他的远程数据库中输入此退货信息，并将更改复制到仓库的统一数据库，从而使 Quantity 列的数字增加了 40（变为 68）。

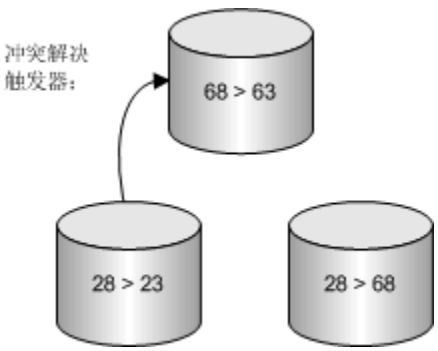


仓库的输入数据被添加到数据库中：Quantity 列现在显示库存中有 68 件小号无袖紧身 T 恤衫。当来自第一个销售代表的更新到达时，便会导致一个冲突—SQL Anywhere 检测到更新是从 28 减到 23，但是该列的当前值是 68。

缺省情况下，最新的更新会成功，库存量将被设置为不正确的值 23。



在本示例中，应该这样解决冲突：将更改变动累加到库存列以生成结果，从而将最终值 63 放置到数据库中。



实现解决方案

适用于此情况的 RESOLVE UPDATE 触发器将添加来自两个更新的增量。例如：

```
CREATE TRIGGER resolve_quantity
 RESOLVE UPDATE OF Quantity
 ON "DBA".Products
 REFERENCING OLD AS old_name
  NEW AS new_name
  REMOTE AS remote_name
 FOR EACH ROW
 BEGIN
   SET new_name.Quantity = new_name.Quantity
                         + old_name.Quantity
                         - remote_name.Quantity
 END;
```

在执行 UPDATE 语句之前，此触发器将统一数据库中的旧值 (68) 与最初的 UPDATE 语句执行时远程数据库中的旧值 (28) 之间的差值与要发送的新值相加。这样，new_name.Quantity 变为 63 (= 23 + 68 - 28)，此值被输入到 Quantity 列中。

远程数据库的一致性是通过以下方式保持的：

1. 原始远程 UPDATE 语句将值从 28 更改为 23。
2. 仓库的输入被复制到远程数据库，但是由于旧值不是预期的值，所以复制将失败。
3. RESOLVE UPDATE 触发器进行的更改被复制到远程数据库。

未找到行错误

某用户删除某行（具有给定主键值）。第二个用户在另一站点更新或删除同一行。在这种情况下，第二个语句失败，因为找不到该行。

要正确复制 UPDATE 和 DELETE 语句，必须将所有主键列包括在项目中。

复制 UPDATE 或 DELETE 语句时，SQL Remote 使用主键列唯一标识要更新或删除的行。所有被复制的表都必须具有一个已声明的主键或唯一约束。仅有唯一的索引是不够的。

WHERE 子句和主键

主键列用于复制的 UPDATE 和 DELETE 语句的 WHERE 子句中。当表没有主键时，WHERE 子句将引用表中的所有列。请参见“复制 INSERT 和 DELETE 语句”一节第 36 页。

另请参见

- “报告并处理复制错误”一节第 123 页

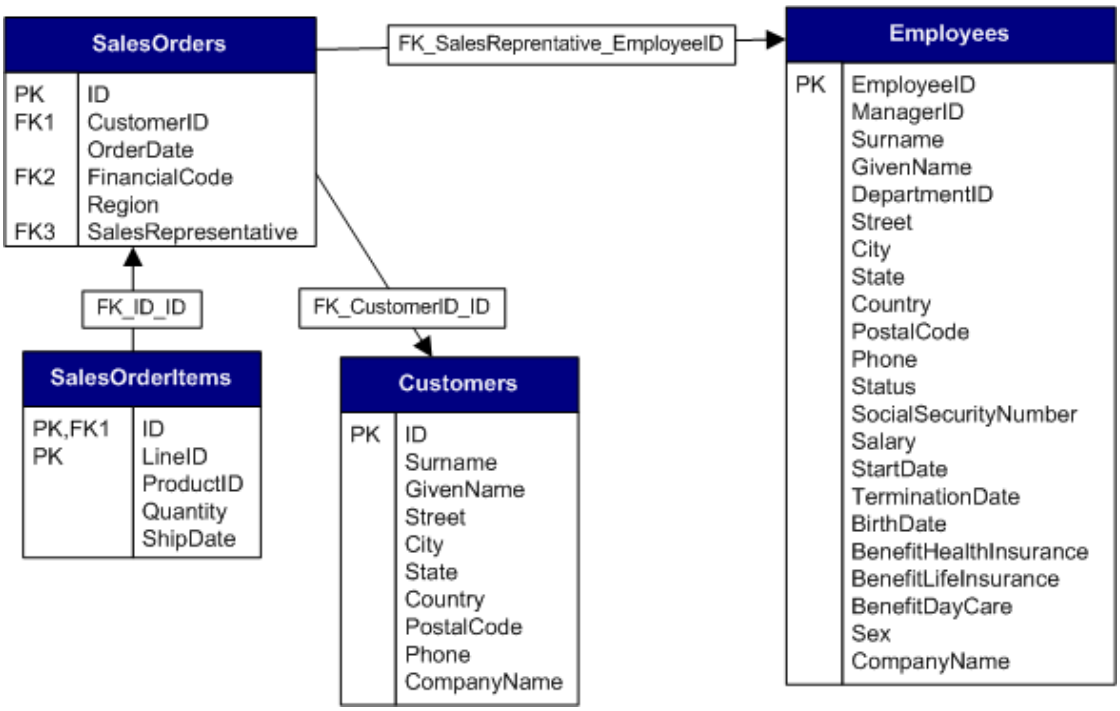
参照完整性错误

关系数据库中的表通常是通过外键引用进行关联的。因此，参照完整性约束确保了数据库保持一致。请参见“实施实体完整性和参照完整性”一节 《SQL Anywhere 服务器 - SQL 的用法》。

只复制数据库的一部分时，必须确保复制的数据库仍然具有参照完整性。

最好避免未复制的引用表错误。您的远程数据库不应包含指向未复制表的外键。

例如，在统一数据库中，SalesOrders 表具有一个指向 Employees 表的外键。SalesOrders.SalesRepresentative 是引用主键 Employees.EmployeeID 的外键。



创建了一个排除 Employees 表、但包括整个 SalesOrder 表的发布：PubSales。

```
CREATE PUBLICATION PubSales (
   TABLE Customers,
   TABLE SalesOrders,
   TABLE SalesOrderItems,
 );
```

某远程用户 Rep1 预订了 PubSales 发布。然后，您从统一数据库中抽取 Rep1 并尝试为 Rep1 创建一个数据库。但是，数据库创建失败，因为 Rep1 缺失 Employees 表。要避免出现此问题，您可以：

- **删除外键引用** 要排除外键引用，请在使用抽取实用程序 (dbxtract) 时指定 -xf 选项。

  但是，如果从远程数据库中删除外键引用，远程数据库中将没有任何约束可以防止将无效的值插入到 SalesOrders 表的 SalesRepresentative 列。

如果远程数据库的 SalesRepresentative 列中插入了无效的值，复制的 INSERT 语句将在统一数据库上失败。

- **在发布中包含缺失的表** 在发布中包含 Employees 表（或至少包含其主键）。例如：

```
CREATE PUBLICATION PubSales (
   TABLE Customers,
   TABLE SalesOrders,
   TABLE SalesOrderItems,
   TABLE Products,
   TABLE Employees
);
```

另请参见

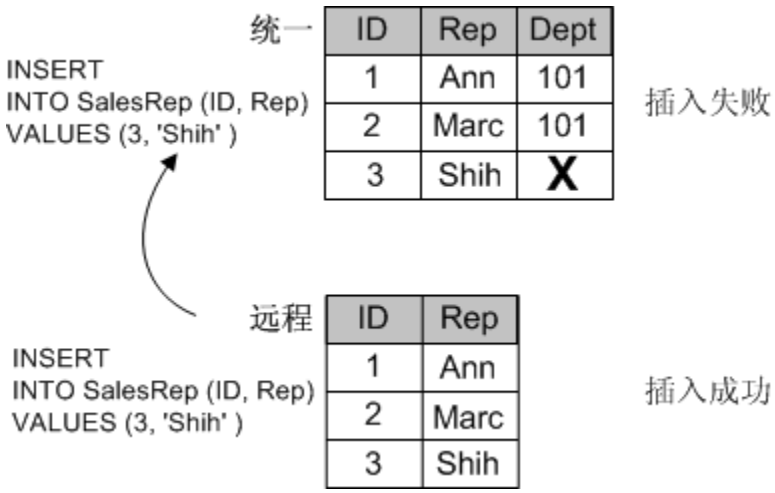
- “报告并处理复制错误”一节第 123 页

插入错误

在将远程数据库中的 INSERT 语句复制到统一数据库时，只能从发布中排除以下类型的列：

- 允许空值的列。
- 有缺省值的列。

如果您排除的列未满足以上其中一个要求，则在远程数据库执行的 INSERT 语句在统一数据库中复制时会失败。



使用 BEFORE 触发器作为解决办法
本示例的一个例外情况是，使用 BEFORE 触发器来维护未包含在 INSERT 语句中的列时。

另请参见

- “复制冲突和错误”一节第 42 页

重复主键错误

当所有用户都连接到同一数据库时，确保每个 INSERT 语句都使用唯一主键是没有问题的。如果某用户尝试重用某个主键，则 INSERT 语句会失败。

复制系统中的情况与此不同，因为用户与多个数据库相连接。当连接到不同远程数据库的两个用户插入使用相同主键值的行时，就会出现潜在的问题。他们的语句都会各自成功，因为主键值在每个远程数据库中是唯一的。

但是，当这两个用户通过相同的统一数据库复制他们自己的数据库时，就会出现问题。第一个要通过统一数据库复制的数据库会成功。但第二个要到达复制系统中给定数据库的插入操作将会失败。

主键值必须是唯一的

要避免主键错误，必须确保在数据库插入某行时，该行的主键在系统的所有数据库中是唯一的。有多种方法可以实现此目标，包括：

1. 使用 SQL Anywhere 数据库的缺省全局自动增量功能。请参见“全局自动增量列”一节 第 53 页。
2. 使用主键池在每个站点维护一个未使用的唯一主键值的列表。请参见“使用主键池”一节 第 54 页。

可以单独使用上述方法，也可以同时使用上述方法来避免出现重复主键。

另请参见

- “报告并处理复制错误”一节第 123 页

全局自动增量列

使用 GLOBAL AUTOINCREMENT 缺省值为每个远程数据库指派一个唯一的全局数据库标识号。

如果您为某列指定了 GLOBAL AUTOINCREMENT 缺省值，则将对该列的值域进行分区。每个分区都包含相同数目的值。例如，如果将数据库中一个整数列的分区大小设置为 1000，则一个分区的范围是从 1001 到 2000，下一个分区从 2001 到 3000，依此类推。

SQL Anywhere 只从用该数据库编号唯一标识的分区提供数据库中的缺省值。例如，如果您为某远程数据库指派了标识号 10，则该数据库中的缺省值将从 10001-11000 这个范围中选择。另一个远程数据库被指派了标识号 11，它将为同一列提供 11001-12000 范围中的缺省值。

另请参见

- “GLOBAL AUTOINCREMENT 缺省值”一节 《SQL Anywhere 服务器 - SQL 的用法》

声明 DEFAULT GLOBAL AUTOINCREMENT

可以通过在 Sybase Central 中选择列属性，或在 CREATE TABLE 或 ALTER TABLE 语句中包含 DEFAULT GLOBAL AUTOINCREMENT 子句在数据库中设置缺省值。

分区大小

或者，也可以在紧随 AUTOINCREMENT 关键字之后的括号中指定分区大小。分区大小可以是任意非负整数，但分区大小的选择一般需要保证任何一个分区内的编号资源很少被用尽。

对于 INT 或 UNSIGNED INT 类型的列，缺省分区大小是 $2^{16} = 65536$；对于其它类型的列，缺省分区大小是 $2^{32} = 4294967296$。由于这些缺省值可能不合适（尤其当列不是 INT 或 BIGINT 类型时），因此建议您显式地指定分区大小。

示例

以下语句创建一个包含两列的表：用于保存客户标识号的整数列和用于保存客户名称的字符串列。标识号列 ID 使用 GLOBAL AUTOINCREMENT 缺省值，其分区大小为 5000。

```
CREATE TABLE Customers (
   ID   INT  DEFAULT GLOBAL AUTOINCREMENT (5000),
   name VARCHAR(128) NOT NULL,
   PRIMARY KEY (ID)
);
```

另请参见

- “CREATE TABLE 语句”一节 《SQL Anywhere 服务器 - SQL 参考》
- “ALTER TABLE 语句”一节 《SQL Anywhere 服务器 - SQL 参考》

使用主键池

主键池是包含 SQL Remote 系统中每个数据库的一组主键值的表。主主键池表在统一数据库中创建及存储。远程用户可通过预订统一数据库主键池表来接收他们自己的那组主键值。当远程用户向表中插入新行时，他们使用存储过程从他们的池中选择有效的主键。该池通过在统一数据库中定期运行一个补充主键源的过程来维护。

主键池技术需要以下组件：

- **主键池表** 在统一数据库中，需要有一个用于保存系统中每个数据库的有效主键值的表。请参见“创建主键池表”一节第 54 页。
- **补充过程** 在统一数据库中，需要有一个用于补充键池表的存储过程。请参见“填充和补充键池”一节第 56 页。
- **键池的共享** 系统中的每个远程数据库都必须从统一数据库的键池表预订自己的那组有效值。请参见“复制主键池”一节第 55 页。
- **数据输入过程** 在远程数据库中，使用某存储过程输入新行，该存储过程会从池中选取下一个有效主键值，然后从键池中删除该值。请参见“使用键池中的主键”一节第 57 页。

创建主键池表

♦ **创建主键池表 (SQL)**

1. 在统一数据库中，执行以下语句创建一个主键池表：


```
CREATE TABLE KeyPool (
   table_name VARCHAR(128) NOT NULL,
   value INTEGER NOT NULL,
   location CHAR(12) NOT NULL,
   PRIMARY KEY (table_name, value),
);
```

列	说明
table_name	保存必须为其维护主键池的表的名称。例如，如果只在统一数据库添加新销售代表，则只有 Customers 表需要主键池，此列是多余的。
value	保存一个主键值列表。每个值对于在 table_name 中列出的每个表都是唯一的。
location	接收者的标识符。在某些系统中，这可能与 SalesReps 表的 rep_key 值相同。在其它系统中，还会有销售代表之外的用户；在这样的系统中，两个标识符应该不同。

2. 要提高性能，可执行以下语句在主键表上创建一个索引：

```
CREATE INDEX KeyPoolLocation
ON KeyPool (table_name, location, value);
```

复制主键池

可以将主键池并入到现有发布中，或将其作为单独的发布共享。使用以下过程为主键池创建一个单独的发布，并为用户预订该发布。

♦ 复制主键池 (SQL)

1. 在统一数据库中，为主键池数据创建一个发布。

```
CREATE PUBLICATION KeyPoolData (
   TABLE KeyPool SUBSCRIBE BY location
);
```

2. 为每个远程数据库创建对 KeyPoolData 发布的预订。

```
CREATE SUBSCRIPTION
  TO KeyPoolData( 'user1' )
  FOR user1;
CREATE SUBSCRIPTION
  TO KeyPoolData( 'user2' )
  FOR user2;
...
```

预订参数是 location 标识符。

另请参见

- “CREATE PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “CREATE SUBSCRIPTION 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》

填充和补充键池

每当远程用户添加一个新客户时，远程用户的可用主键池中的主键就会减少 1 个。因此，您需要定期补充统一数据库的主键池表中的内容，然后将新的主键复制到远程数据库。

♦ 最初填充主键池 (SQL)

1. 在统一数据库中，创建一个过程来填充主键池。

   不要使用触发器来补充键池
   因为不会复制触发器操作，所以不能使用触发器来补充键池。

   例如：

```
CREATE PROCEDURE ReplenishPool()
BEGIN
   FOR EachTable AS TableCursor
   CURSOR FOR
      SELECT table_name
      AS CurrTable, max(value) as MaxValue
      FROM KeyPool
      GROUP BY table_name
   DO
      FOR EachRep AS RepCursor
      CURSOR FOR
         SELECT location
         AS CurrRep, COUNT(*) AS NumValues
         FROM KeyPool
         WHERE table_name = CurrTable
         GROUP BY location
      DO
         // make sure there are 100 values.
         // Fit the top-up value to your
         // requirements
         WHILE NumValues < 100 LOOP
            SET MaxValue = MaxValue + 1;
            SET NumValues = NumValues + 1;
            INSERT INTO KeyPool
            (table_name, location, value)
            VALUES
            (CurrTable, CurrRep, MaxValue);
         END LOOP;
      END FOR;
   END FOR;
END;
```

2. 在主键池中，为每个用户插入一个初始主键值。

   ReplenishPool 过程要求对于每个预订者都至少存在一个主键值，这样它可以找到最大值并加 1 以生成下一个集合。

   要实现池的初始填充，可以为每个用户插入单个值，然后调用 ReplenishPool 来填充其余部分。以下示例针对三个远程用户和一个名为 Office 的统一用户说明了此过程：

```
INSERT INTO KeyPool VALUES( 'Customers', 40, 'user1' );
INSERT INTO KeyPool VALUES( 'Customers', 41, 'user2' );
INSERT INTO KeyPool VALUES( 'Customers', 42, 'user3' );
INSERT INTO KeyPool VALUES( 'Customers', 43, 'Office');
CALL ReplenishPool();
```


ReplenishPool 过程会为每个用户将池一直填充到 100 个值。所需要的值取决于用户在数据库的表中插入行的频率。

3. 定期运行 ReplenishPool。

   必须定期在统一数据库运行 ReplenishPool 过程以补充 KeyPool 表中主键值的池。

使用键池中的主键

当销售代表向 Customers 表添加新客户时，使用一个存储过程获取要插入的主键值。此示例使用一个存储过程来提供主键值，使用另一个存储过程来执行插入操作。

♦ 使用主键 (SQL)

1. 创建一个在远程数据库中运行的过程以从主键池表中获取主键。

   例如，NewKey 过程提供键池中的一个整数值，并将该值从池中删除。

```
CREATE PROCEDURE NewKey(
      IN @table_name VARCHAR(40),
      OUT @value INTEGER )
BEGIN
   DECLARE NumValues INTEGER;

   SELECT COUNT(*), MIN(value)
   INTO NumValues, @value
         FROM KeyPool
         WHERE table_name = @table_name
         AND location = CURRENT PUBLISHER;
   IF NumValues > 1 THEN
      DELETE FROM KeyPool
      WHERE table_name = @table_name
      AND value = @value;
   ELSE
   // Never take the last value, because
   // ReplenishPool will not work.
   // The key pool should be kept large enough
   // that this never happens.
      SET @value = NULL;
   END IF;
END;
```

   NewKey 过程利用这样一个事实：销售代表标识符是远程数据库的 CURRENT PUBLISHER。

2. 创建一个在远程数据库中运行的过程以在预订表中插入新行。

   例如，NewCustomers 过程将新客户插入到表中，同时使用通过 NewKey 获取的值来构造主键。

```
CREATE PROCEDURE NewCustomers(
      IN customer_name CHAR( 40 ) )
BEGIN
   DECLARE new_cust_key INTEGER ;
   CALL NewKey( 'Customers', new_cust_key );
   INSERT
   INTO Customers (
      cust_key,
      name,
      location
      )
```

```
    VALUES (
       'Customers ' ||
       CONVERT (CHAR(3), new_cust_key),
       customer_name,
       CURRENT PUBLISHER
       );
);
END
```

可通过这种方式增强此过程：对通过 NewKey 获取的 new_cust_key 值进行测试以检查该值是否为 NULL，如果是 NULL，则禁止插入。

在远程数据库之间对行进行分区

每个远程数据库都可以包含存储在统一数据库中的另一个数据子集。可创建您的发布和预订，以便在远程数据库之间对数据进行分区。

分区可以是无交集的，也可以包含交集。例如，如果每个员工都拥有其自己的客户集，而没有共享的客户，则分区将是**无交集**的。如果存在共享客户，这些客户出现在多个远程数据库中，则分区中会包含**交集**。

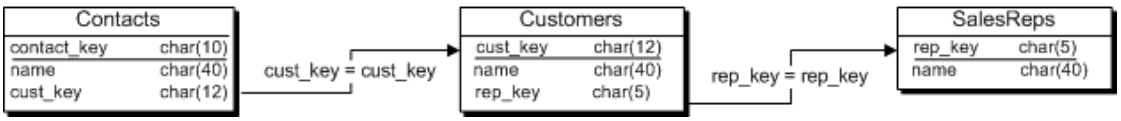
有时，即使当表中不存在预订表达式时，也需要对表的行进行分区。请参见“使用无交集数据分区”一节第 60 页。

有时，当数据库中存在多对多关系时，也需要对表进行分区。请参见“使用有交集分区”一节第 65 页。

使用无交集数据分区

当远程数据库不共享数据时，数据分区是无交集的。例如，每个销售代表都有自己的一组客户，并且他们不与其他销售代表共享客户。

在下面的示例中，有三个表存储着销售代表和客户之间的交互信息：Customers、Contacts 和 SalesReps。每个销售代表都向若干个客户销售产品。某些客户只有一个联系人，而另一些客户则有多个联系人。



Contacts、Customers 和 SalesReps 表的说明

下表介绍了 Customers、Contacts 和 SalesReps 数据库表，详见“使用无交集数据分区”一节 第 60 页。

表	说明	表定义
Contacts	与该公司有业务来往的所有单独联系人。每个联系人都属于单个客户。Contacts 表包含以下几列： ● **contact_key** 每个联系人的标识符。这是主键。 ● **name** 每个联系人的姓名。 ● **cust_key** 该联系人所属客户的标识符。这是一个指向 Customers 表的外键。	`CREATE TABLE Contacts ( contact_key CHAR(12) NOT NULL, name CHAR(40) NOT NULL, cust_key CHAR(12) NOT NULL, FOREIGN KEY REFERENCES Customers, PRIMARY KEY (contact_key) );`
Customers	与该公司有业务来往的所有客户。Customers 表包含以下几列： ● **cust_key** 每个客户的标识符。这是主键。 ● **name** 每个客户的名称。 ● **rep_key** 某销售关系中销售代表的标识符。这是一个指向 SalesReps 表的外键。	`CREATE TABLE Customers ( cust_key CHAR(12) NOT NULL, name CHAR(40) NOT NULL, rep_key CHAR(12) NOT NULL, FOREIGN KEY REFERENCES SalesReps, PRIMARY KEY (cust_key) );`
SalesReps	在该公司工作的所有销售代表。SalesReps 表包含以下几列： ● **rep_key** 每个销售代表的标识符。这是主键。 ● **name** 每个销售代表的姓名。	`CREATE TABLE SalesReps ( rep_key CHAR(12) NOT NULL, name CHAR(40) NOT NULL, PRIMARY KEY (rep_key) );`

销售代表必须预订提供以下信息的发布：

- **为公司工作的所有销售代表的列表** 以下语句将创建一个发布整个 SalesRep 表的发布：

```
CREATE PUBLICATION SalesRepData (
   Table SalesReps ...)
);
```

- **指派给他们的客户列表** 此信息可在 Customers 表中找到。以下语句将创建一个发布 Customers 表的发布，该发布包含与 Customers 表中的 rep_key 列值匹配的行：

```
CREATE PUBLICATION SalesRepData (
   TABLE Customers SUBSCRIBE BY rep_key ...
);
```

- **他们的被指派客户的联系信息列表** 此信息可在 Contacts 表中找到。Contacts 表必须在销售代表之间进行分区，但没有对 SalesRep 表中的 rep_key 值的任何引用。要解决此问题，可以在 Contacts 项目中使用一个引用 Customers 表的 rep_key 列的子查询。

  以下语句将创建一个发布 Contacts 表的发布，该发布包含引用 Customers 表的 rep_key 列的行。

```
CREATE PUBLICATION SalesRepData ( ...
   TABLE Contacts
      SUBSCRIBE BY (SELECT rep_key
         FROM Customers
         WHERE Contacts.cust_key = Customers.cust_key )
);
```

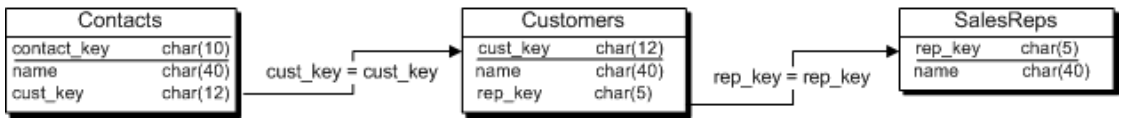
  Customers 表中的某一行含有 Contacts 表的当前行中的 cust_key 值；SUBSCRIBE BY 语句中的 WHERE 子句可确保该子查询只返回一个值。

以下语句创建完整的发布：

```
CREATE PUBLICATION SalesRepData (
   TABLE SalesReps
   TABLE Customers
      SUBSCRIBE BY rep_key
   TABLE Contacts
      SUBSCRIBE BY (SELECT rep_key
         FROM Customers
         WHERE Contacts.cust_key = Customers.cust_key )
);
```

使用 BEFORE UDPATE 触发器

在下面的示例中，有三个表存储着销售代表和客户之间的交互信息：Customers、Contacts 和 SalesReps。每个销售代表都向若干个客户销售产品。某些客户只有一个联系人，而另一些客户则有多个联系人。



有关这些表的详细说明，请参见“Contacts、Customers 和 SalesReps 表的说明”一节第 60 页。

一个销售代表预订了一个发布，该发布提供了一份 SalesRep 表、一份含有指派给这些销售代表的客户的详细信息的 Customers 表以及一份含有对应于他们的客户的联系人详细信息的 Contacts 表。例如，每个销售代表都预订了以下的发布：

```
CREATE PUBLICATION SalesRepData (
   TABLE SalesReps
   TABLE Customers
      SUBSCRIBE BY rep_key
   TABLE Contacts
      SUBSCRIBE BY (SELECT rep_key
         FROM Customers
         WHERE Contacts.cust_key = Customers.cust_key )
);
```

有关此发布的详细说明，请参见“使用无交集数据分区”一节第 60 页。

保持参照完整性

这种在预订者之间重新指派行的做法有时称为**地域调整**，因为它是销售人员自动应用程序的常见功能，在这种功能中客户被定期重新指派给销售代表。

在统一数据库中，将某客户重新指派给新的销售代表时，就会相应更新 Customers 表中的 rep_key 值。

以下语句会将客户 cust1 重新指派给另一个销售代表 rep2。

```
UPDATE Customers
SET rep_key = 'rep2'
WHERE cust_key = 'cust1';
```

此更新以下面的方式被复制：

- 作为一个 DELETE 语句复制到以前的销售代表的远程数据库的 Customers 表中。
- 作为一个 INSERT 语句复制到新销售代表的远程数据库的 Customers 表中。

Contacts 表**不**发生改变。在统一数据库的事务日志中没有关于 Contacts 表的条目。这样，远程数据库中的 SQL Remote 无法重新指派 Contacts 表的 cust_key 行。这将导致产生以下参照完整性问题：以前的销售代表的远程数据库中的 Contacts 表包含一个不再有客户的 cust_key 值。

解决方法是使用 BEFORE UPDATE 触发器。BEFORE UPDATE 触发器不会对数据库表进行任何更改，但会在统一数据库的事务日志中创建一个条目。

触发 BEFORE UPDATE 触发器必须满足以下条件：

- 在运行 UPDATE 语句前触发。以便计算行的 BEFORE 值并将其添加到事务日志。
- 针对每行，而不是针对每个语句。由触发器提供的信息必须是新的预订表达式。

例如，下面的语句将创建一个 BEFORE UPDATE 触发器。

```
CREATE TRIGGER "UpdateCustomer" BEFORE UPDATE OF "rep_key"
// only fire the trigger when we modify rep_key, not any other column
ORDER 1 ON "Cons"."Customers"
/* REFERENCING OLD AS old_name NEW AS new_name */
REFERENCING NEW AS NewRow
   OLD AS OldRow
FOR EACH ROW
```



```
BEGIN
// determine the new subscription expression
   // for the Customers table
     UPDATE Contacts
   PUBLICATION SalesRepData
   OLD SUBSCRIBE BY ( OldRow.rep_key )
   NEW SUBSCRIBE BY ( NewRow.rep_key )
   WHERE cust_key = NewRow.cust_key;
END
;
```

SQL Remote 使用事务日志中的信息来确定哪些预订者接收哪些行。

执行了该语句后，统一数据库事务日志中将包含两个条目：

- 由 BEFORE UPDATE 触发器生成的用于 Contacts 表的 INSERT 和 DELETE 语句。

```
--BEGIN TRIGGER-1029-0000461705
--BEGIN TRANSACTION-1029-0000461708
BEGIN TRANSACTION
go
--UPDATE PUBLICATION-1029-0000461711 Cons.Contacts
--PUBLICATION-1029-0000461711-0002-NEW_SUBSCRIBE_BY-rep2
--PUBLICATION-1029-0000461711-0002-OLD_SUBSCRIBE_BY-rep1
--NEW-1029-0000461711
--INSERT INTO Cons.Contacts(contact_key,name,cust_key)
--VALUES ('5','Joe','cust1')
go
--OLD-1029-0000461711
--DELETE FROM Cons.Contacts
-- WHERE contact_key='5'
go
--END TRIGGER-1029-0000461743
```

- 执行的原始 UPDATE 语句以及分别获得行或丢失行的这些用户的 INSERT 和 DELETE 语句。

```
--PUBLICATION-1029-0000461746-0002-NEW_SUBSCRIBE_BY-rep2
--PUBLICATION-1029-0000461746-0002-OLD_SUBSCRIBE_BY-rep1
--NEW-1029-0000461746
--INSERT INTO Cons.Customers(cust_key,name,rep_key)
--VALUES ('cust1','company1','rep2')
go
--OLD-1029-0000461746
--DELETE FROM Cons.Customers
-- WHERE cust_key='cust1'
go
--UPDATE-1029-0000461746
UPDATE Cons.Customers
   SET rep_key='rep2'
VERIFY (rep_key)
VALUES ('1')
 WHERE cust_key='cust1'
go
--COMMIT-1029-0000461785
COMMIT WORK
```

SQL Remote 会对事务日志进行扫描以查找 BEFORE 和 AFTER 标记。根据此信息，它可以确定哪些远程用户得到了 INSERT、UPDATE 或 DELETE 语句。

- 当用户在 BEFORE 列表中而不在 AFTER 列表中时，会将 DELETE 语句发送到 Contacts 表中。
- 当用户在 AFTER 列表中而不在 BEFORE 列表中时，会将 INSERT 语句发送到 Contacts 表中。
- 如果用户既在 BEFORE 列表中又在 AFTER 列表中，则不会对 Contacts 表执行任何操作，但是会发送 Customers 表中的 UPDATE 语句。

如果 BEFORE 和 AFTER 列表相同，则远程用户已具有该行，并将发送 UPDATE 语句。

有关触发器的说明

在以下示例中，必须使用 BEFORE UPDATE 触发器。在其它上下文中，BEFORE DELETE 和 BEFORE INSERT 都是必需的。

```
UPDATE table-name
PUBLICATION pub-name
   SUBSCRIBE BY sub-expression
WHERE search-condition;
```

在本示例中，使用 BEFORE 触发器。

```
UPDATE table-name
PUBLICATION publication-name
   OLD SUBSCRIBE BY old-subscription-expression
   NEW SUBSCRIBE BY new-subscription-expression
WHERE search-condition;
```

UPDATE 语句列出了受影响的发布和表。语句中的 WHERE 子句说明了受影响的行。此 UPDATE 语句没有更改表中的数据，它只在事务日志中生成条目。

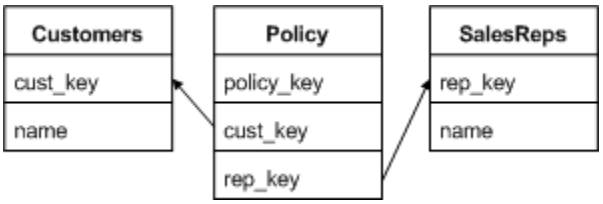
在本示例中，预订表达式返回单个值。但也可以使用返回多个值的子查询。预订表达式的值必须是更新后的值。

在本示例中，该行的唯一预订者是新销售代表。有关具有现有预订者和新预订者的行的示例，请参见“使用有交集分区”一节第 65 页。

使用有交集分区

当远程数据库共享数据时，数据分区是有交集的。例如，销售代表相互之间就共享客户。

假设有三个表存储着销售代表和客户之间的交互信息：Customers、Policy 和 SalesReps。每个销售代表都向若干个客户销售产品，并且某些客户与多个销售代表都有业务来往。Policy 表具有指向 Customers 表和 SalesReps 表的外键。在 Customers 和 SalesReps 之间存在多对多的关系。



Customers、Policy 和 SalesReps 表的说明

下表介绍了 Customers、Policy 和 SalesReps 数据库表，详见“使用有交集分区”一节第 65 页。

表	说明

Customers	与该公司有业务来往的所有客户。Customers 表包含以下各列：
	● **cust_key** 包含每个客户的标识符的主键列。
	● **name** 包含每个客户的名称的列。
	可使用以下语句创建此表：
	`CREATE TABLE Customers (`
	`  cust_key CHAR(12) NOT NULL,`
	`  name CHAR(40) NOT NULL,`
	`  PRIMARY KEY (cust_key)`
	`);`

表	说明

Policy	一个由三列组成的表，用于维护客户与销售代表间的多对多关系。Policy 表包含以下几列：
	● **policy_key** 包含销售关系标识符的主键列。
	● **cust_key** 包含销售关系中客户代表的外键的列。
	● **rep_key** 包含销售关系中销售代表的外键的列。
	可使用以下语句创建此表：
	`CREATE TABLE Policy (`
	`  policy_key CHAR(12) NOT NULL,`
	`  cust_key CHAR(12) NOT NULL,`
	`  rep_key CHAR(12) NOT NULL,`
	`  FOREIGN KEY ( cust_key )`
	`  REFERENCES Customers ( cust_key )`
	`  FOREIGN KEY ( rep_key )`
	`  REFERENCES SalesReps (rep_key ),`
	`  PRIMARY KEY ( policy_key )`
	`);`

SalesReps	在该公司工作的所有销售代表。SalesReps 表包含以下几列：
	● **rep_key** 每个销售代表的标识符。这是主键。
	● **name** 每个销售代表的姓名。
	可使用以下语句创建此表：
	`CREATE TABLE SalesReps (`
	`  rep_key CHAR(12) NOT NULL,`
	`  name CHAR(40) NOT NULL,`
	`  PRIMARY KEY (rep_key)`
	`);`

数据分区

客户和销售代表之间的多对多关系对正确共享信息提出了新的挑战。

销售代表必须预订提供以下信息的发布：

- **整个 SalesReps 表** 此项目没有限定符，因此发布中包含整个 SalesReps 表。

```
...
   TABLE SalesReps,
...
```

- **Policy 表中那些所包含的销售关系涉及到预订该数据的销售代表的行** 该项目使用一个 SUBSCRIBE BY 预订表达式来指定用于在销售代表间对数据进行分区的列：

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


该预订表达式可确保每个销售代表只接收表中那些 rep_key 列的值与预订中提供的值匹配的行。

Policy 表分区**无交集**：不存在任何由多个预订者共享的行。

- **Customers 表中那些列出与预订数据的销售代表有业务来往的客户的行** Customers 表没有引用在预订中使用的对数据进行分区的销售代表值。此问题可以通过在发布中使用子查询得到解决。

  Customers 表中的每一行都可能与 SalesReps 表中的多个行相关，并被多个销售代表的数据库共享。也就是说，存在重叠预订。

  含有子查询的预订表达式用于定义分区。该项目定义如下：

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

  Customers 分区是**相交的**：某些行由多个预订者共享。

以下语句创建完整的发布：

```
CREATE PUBLICATION SalesRepData (
   TABLE SalesReps,
   TABLE Policy SUBSCRIBE BY rep_key,
   TABLE Customers SUBSCRIBE BY (
      SELECT rep_key FROM Policy
      WHERE Policy.cust_key =
         Customers.cust_key
   )
);
```

发布中的多值子查询

Customers 项目中的子查询在其结果集中返回单个列 (rep_key)，但可能返回多个行，这些行对应于那些与特定客户有业务来往的所有销售代表。如果某个预订表达式具有多个值，则会将行复制给其预订与这些值中的任意一个值相匹配的所有预订者。这种具有多值预订表达式的可能性实现了对表的有交集分区。

在预订者之间重新指派行时保持参照完整性

若要取消某客户与某销售代表间的销售关系，应删除 Policy 表中的某行。在本示例中，Policy 表更改被正确复制给以前的销售代表。但未对 Customers 表进行任何更改，因此也没有将 Customers 表的任何更改复制给以前的销售代表。

在没有触发器的情况下，这将使预订者在 Customers 表中留下不正确的数据。在将新行添加到 Policy 表时也会产生此类问题。

使用触发器解决该问题

解决方法是编写可在 Policy 表被更改时触发的 BEFORE 触发器。这些特殊的触发器不会对数据库表进行任何更改，但会在 SQL Remote 用于维护预订者数据库中的数据的事务日志中创建一个条目。

BEFORE INSERT 触发器

例如，以下语句创建一个 BEFORE INSERT 触发器，该触发器可跟踪对 Policy 表执行的插入操作并确保远程数据库包含正确的数据。

```
CREATE TRIGGER InsPolicy
BEFORE INSERT ON Policy
REFERENCING NEW AS NewRow
FOR EACH ROW
BEGIN
   UPDATE Customers
   PUBLICATION SalesRepData
   SUBSCRIBE BY (
      SELECT rep_key
      FROM Policy
      WHERE cust_key = NewRow.cust_key
      UNION ALL
      SELECT NewRow.rep_key
   )
   WHERE cust_key = NewRow.cust_key;
END;
```

BEFORE DELETE 触发器

以下语句创建一个 BEFORE DELETE 触发器，该触发器可跟踪对 Policy 表执行的删除操作：

```
CREATE TRIGGER DelPolicy
BEFORE DELETE ON Policy
REFERENCING OLD AS OldRow
FOR EACH ROW
BEGIN
   UPDATE Customers
   PUBLICATION SalesRepData
   SUBSCRIBE BY (
      SELECT rep_key
      FROM Policy
      WHERE cust_key = OldRow.cust_key
      AND Policy_key <> OldRow.Policy_key
   )
   WHERE cust_key = OldRow.cust_key;
END;
```

UPDATE PUBLICATION 语句的 SUBSCRIBE BY 子句包含一个子查询，该子查询可以是多值的。

多值子查询

UPDATE PUBLICATION 语句的 SUBSCRIBE 子句中的子查询是一个 UNION 表达式，并且该子查询可以是多值的：

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

- UNION 的第一部分是与该客户有业务关系的现有销售代表的集合，这是从 Policy 表获取的。

  预订查询的结果集肯定是接收该行的所有销售代表，而不仅仅是新销售代表。

- UNION 的第二部分是与该客户有业务来往的新销售代表的 rep_key 值，该值是从 INSERT 语句获取的。

BEFORE DELETE 触发器中的子查询是多值的：

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

- 该子查询从 Policy 表中获取 rep_key 值。这些值包含与要转移的客户有业务来往的所有销售代表的主键值 (WHERE **cust_key** = **OldRow.cust_key**)，要删除的销售代表除外 (AND **rep_key** <> **OldRow.rep_key**)。

  预订查询的结果集肯定是执行删除操作后所有与接收该行的销售代表匹配的值。

注意

- Customers 表中的数据不是通过单个预订者（例如通过主键值）标识的，而是由多个预订者共享。这可能导致数据在多个远程站点上的各复制消息之间更新，进而造成复制冲突。解决此问题的方法是，通过权限解决（例如，只授予某些用户更新 Customers 表的权限），或者是向数据库添加 RESOLVE UPDATE 触发器以按编程方式处理冲突。
- 这里未介绍对 Policy 表的更新。应避免进行这些更新，或必须创建一个结合了示例中介绍的 BEFORE INSERT 和 BEFORE DELETE 两个触发器功能的 BEFORE UPDATE 触发器。

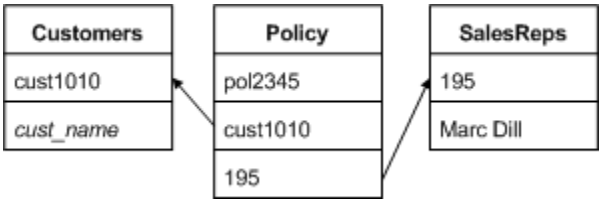
对多对多关系使用 subscribe_by_remote 选项

当 subscribe_by_remote 选项设置为 On 时，来自远程数据库的对 SUBSCRIBE BY 值为 NULL 或空字符串的行执行的操作会假定远程用户预订了该行。缺省情况下，subscribe_by_remote 选项设置为 On。

subscribe_by_remote 选项解决了在不使用该选项时某些发布可能会引发的问题。以下发布对 Customers 表预订表达式使用一个子查询，这是因为客户可能与若干个销售代表相关：

```
CREATE PUBLICATION SalesRepData (
   TABLE SalesReps,
   TABLE Policy SUBSCRIBE BY rep_key,
   TABLE Customers SUBSCRIBE BY (
      SELECT rep_key FROM Policy
      WHERE Policy.cust_key =
         Customers.cust_key
   ),
);
```

例如，Marc Dill 是一个销售代表，他刚刚和一个新客户商定了一个策略。他在 Customers 表中插入一个新行并在 Policy 表中也插入一行以将这个新客户指派给他自己。



在统一数据库中，SQL Remote 会执行 Customers 行的插入操作，SQL Anywhere 会在插入操作执行时在事务日志中记录下预订值。

随后，当 SQL Remote 对事务日志进行扫描时，会从预订表达式生成一个预订者列表，但 Marc Dill 不在该列表中，因为还未应用 Policy 表中将客户指派给他的那个行。如果 subscribe_by_remote 被设置为 Off，则结果是将这个新客户作为一个 DELETE 语句发送回 Marc Dill。

只要 subscribe_by_remote 设置为 On，SQL Remote 就会假定该行与插入它的销售代表相关联，也不会将 INSERT 复制回 Marc Dill，并且复制系统保持原样。

如果 subscribe_by_remote 设置为 Off，则您必须确保 Policy 行在 Customers 行之前插入，并通过将检查推迟到事务结束来避免参照完整性违规。

另请参见

- “subscribe_by_remote 选项 [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》

为每个数据库指派唯一的标识号

必须为每个远程数据库指派一个不同的标识号。您可以通过多种方法来创建和分发标识号。一种方法是将这些值放置在一个表中，然后根据其它一些唯一属性（例如用户名）将该表中相应的行下载到每个数据库。

使用 global_database_id 选项

每个数据库中的公共选项 global_database_id 都必须设置为一个唯一的非负整数。某一特定数据库的缺省值范围是从 *pn* + 1 到 *p*(*n* + 1)，其中 *p* 为分区大小，而 *n* 为公共选项 global_database_id 的值。例如，如果分区大小为 1000 而 global_database_id 设置为 3，则范围就是从 3001 到 4000。

当 global_database_id 设置为非负整数时，SQL Anywhere 将应用以下规则来选择缺省值：

- 当列不包含当前分区中的任何值时，第一个缺省值为 *pn* + 1。
- 如果列包含当前分区中的值，但都小于 *p*(*n* + 1)，则下一个缺省值为该范围内上一个最大值加 1。
- 缺省列值不受列中当前分区外的值的影响；也就是说，不受小于 *pn* + 1 或大于 *p*(*n* + 1) 的值的影响。如果通过 MobiLink 同步从另一个数据库中复制了这些值，则这些值就可能存在。

如果将公共选项 global_database_id 设置为缺省值 2147483647，则在列中插入空值。如果不允许空值，则插入行的尝试将导致错误。例如，如果列包含在表的主键中，便会发生这种情况。

由于不能将公共选项 global_database_id 设置为负值，因此所选值始终为正。最大标识号仅受列数据类型和分区大小的限制。

当分区内的可用值用完时，也会生成空缺省值。在此示例中，应为数据库指派一个新的 global_database_id 值，以便可以从另一个分区中选择缺省值。如果该列不允许使用空值，则插入空值的尝试将导致错误。若要检测提供的未用值是否过小并处理此情况，请创建一个 GlobalAutoincrement 类型的事件。

如果某特定分区中的值已用尽，您可以为该数据库指派一个新的数据库 ID。您可以通过任何简便的方式指派新的数据库 ID。但是，一种可行的方法是维护未使用的数据库 ID 值的池。该池的维护方式与主键池的维护方式相同。请参见“使用主键池”一节第 54 页。

可以设置一个事件处理程序，在分区快要用尽时自动通知数据库管理员（或执行其它操作）。请参见“定义事件的触发条件”一节 《SQL Anywhere 服务器 - 数据库管理》。

另请参见

- “global_database_id 选项 [数据库]”一节 《SQL Anywhere 服务器 - 数据库管理》

设置 global_database_id 值

♦ **设置全局数据库标识号 (SQL)**

- 设置 global_database_id 选项的值。标识号必须是非负整数。

  例如，以下语句将数据库标识号设置为 20。

```
SET OPTION PUBLIC.global_database_id = 10;
```

如果特定列的分区大小为 5000，则在 100001-105000 范围内选择此数据库的缺省值。

另请参见

- “global_database_id 选项 [数据库]”一节 《SQL Anywhere 服务器 - 数据库管理》

抽取数据库时设置唯一数据库标识号

如果使用抽取实用程序 (dbxtract) 或 [**抽取数据库向导**] 创建远程数据库，则可以编写一个存储过程来将设置唯一数据库标识号的任务自动化。

♦ **要使设置唯一数据库标识号自动化**

1. 创建一个名为 **sp_hook_dbxtract_begin** 的存储过程。

   例如，要为 user_id 为 101 的远程用户 user2 抽取一个数据库，可执行以下语句：

```
SET OPTION "PUBLIC"."global_database_id" = '1';
CREATE TABLE extract_id (next_id INTEGER NOT NULL) ;
INSERT INTO extract_id VALUES( 1 );
CREATE PROCEDURE sp_hook_dbxtract_begin
AS
    DECLARE @next_id  INTEGER
    UPDATE extract_id SET next_id = next_id + 1000
    SELECT @next_id = (next_id )
    FROM extract_id
    COMMIT
    UPDATE #hook_dict
    SET VALUE = @next_id
    WHERE NAME = 'extracted_db_global_id';
```

   每个抽取或重新抽取的数据库将得到一个不同的 global_database_id。第一个从 1001 开始，下一个则从 2001 开始，依此类推。

2. 通过 -v 选项运行抽取实用程序 (dbxtract) 或运行 [**抽取数据库向导**] 来抽取远程数据。抽取实用程序会执行以下任务：

   a. 创建一个名为 #hook_dict 的临时表，其内容包括：

name	value
extracted_db_global_id	正在被抽取的用户 ID

   如果您将 sp_hook_dbxtract_begin 过程编写为修改行的 value 列，则该值将用作抽取的数据库的 global_database_id 选项，且标记 DEFAULT GLOBAL AUTOINCREMENT 值的主键值范围的起始值。

   - 未定义 sp_hook_dbxtract_begin 过程时，被抽取的数据库的 global_database_id 将被设置为 101。
   - 如果定义了 sp_hook_dbxtract_begin 过程，但它未修改 #hook_dict 中的任何行，则 global_database_id 仍被设置为 101。

b. 调用 **sp_hook_dbxtract_begin**。

c. 输出以下信息以帮助调试过程挂接：

- 找到的过程挂接。
- 调用过程挂接之前 #hook_dict 的内容。
- 调用过程挂接之后 #hook_dict 的内容。

另请参见

- “#hook_dict 表”一节第 156 页
- “SQL Remote 系统过程”一节第 156 页
- “global_database_id 选项 [数据库]”一节 《SQL Anywhere 服务器 - 数据库管理》

SQL Remote 部署和管理

本节将介绍 SQL Remote 的部署和管理问题。

SQL Remote 管理

目录

SQL Remote 的管理概述

可以从统一数据库部署和管理 SQL Remote 系统。

♦ 部署和管理 SQL Remote 系统

1. 设置统一数据库。

   请参见“创建 SQL Remote 系统”一节第 12 页。

2. 查看和测试您的 SQL Remote 系统。

   部署之前应对 SQL Remote 系统进行彻底的测试，尤其是具有大量远程数据库时，更应如此。

3. 创建远程数据库并部署设计。

   您以具有 DBA 权限的用户身份通过以下操作部署 SQL Remote：

   a. 为每个远程用户创建 SQL Anywhere 数据库（数据库中包含用户自己的初始数据副本），并启动其预订。请参见“抽取远程数据库”一节第 79 页。

   b. 在每个远程用户的计算机上安装 SQL Anywhere 数据库服务器、远程数据库、SQL Remote 和客户端应用程序。请参见“部署嵌入式数据库应用程序”一节 《SQL Anywhere 服务器 - 编程》和“部署 SQL Remote”一节 《SQL Anywhere 服务器 - 编程》。

4. 运行消息代理 (dbremote) 以交换消息。

   要交换消息，需执行以下操作：

   a. 确定在统一数据库和远程数据库上是在连续模式下还是在批处理模式下运行消息代理 (dbremote)。“选择消息代理 (dbremote) 模式”一节第 86 页.

   b. 确保系统的配置正确，具有正确的用户名、消息代理 (dbremote) 连接字符串、权限等。请参见“了解消息代理 (dbremote)”一节第 86 页。

5. 管理消息。

   使用 "保证消息传送系统" 管理正在许多数据库间来回传送的消息。请参见“了解保证消息传送系统”一节第 99 页。

6. 提高性能。

   请参见“提高 SQL Remote 的性能”一节第 91 页。

7. 执行备份和恢复策略。

   必须为统一数据库创建和执行备份与恢复策略。请参见“备份 SQL Remote 系统”一节第 114 页。

8. 处理错误。

   请参见“报告并处理复制错误”一节第 123 页。

9. 根据需要升级软件和数据库模式。

   请参见“升级和重新同步”一节第 129 页。

抽取远程数据库

要为远程用户创建数据库，需从统一数据库**抽取**远程数据库。

可以为指定远程用户使用 [**抽取数据库向导**] 或抽取实用程序 (dbxtract) 来从统一数据库抽取远程数据库。两种方法都允许您执行以下一项或多项任务：

- **自动将模式和数据直接抽取并重装到新的或现有数据库中** 这是可在了解 SQL Remote 时使用的恰当方法。如果使用此方法，则不会在磁盘上创建数据的中间副本。此方法为数据提供了更高的安全性。但是，它需要花费更长的时间来实施。请参见“自动抽取远程数据库”一节第 79 页。
- **将模式和数据抽取到文件中，然后将其装载到新的或现有数据库中** 部署 SQL Remote 时，此方法为首选。可以编辑模式文件来自定义对远程数据库的抽取和创建。请参见“将远程数据库抽取到重装文件”一节第 81 页。

  要提高创建多个远程数据库的效率，请参见“创建多个远程数据库”一节第 84 页。

自动抽取远程数据库

有关自动抽取远程数据库及其替代方法（将远程数据库抽取到重装文件）的信息，请参见“抽取远程数据库”一节第 79 页。

通过以下过程抽取统一数据库并将模式和数据重装到新数据库中。不会在磁盘上创建数据的任何中间副本。

♦ 自动抽取远程数据库 (Sybase Central)

1. 以具有 DBA 权限的用户身份连接到统一数据库。
2. 从 [**工具**] 菜单选择 [**SQL Anywhere 11**] » [**抽取数据库**]。
3. 出现提示时选择 [**抽取并重装到新数据库**]。

   出现提示时选择 [**抽取结构和数据**]。
4. 按照向导中的说明进行操作并接受缺省值。

   具有适当模式、远程用户、发布、预订和触发器的新远程数据库将会创建。缺省情况下，会将统一数据库中的数据抽取到远程数据库，并启动预订。但是，该向导不会启动消息代理，所以没有消息被交换。请参见“了解消息代理 (dbremote)”一节第 86 页。

♦ 自动抽取远程数据库 (SQL)

1. 以具有 DBA 权限的用户身份连接到统一数据库。
2. 运行抽取实用程序 (dbxtract)，并指定 -ac 选项以抽取到现有数据库，或指定 -an 选项以抽取到新的数据库。请参见“抽取实用程序 (dbxtract)”一节第 147 页。

   如果您指定 -an 选项，则必须在运行抽取实用程序 (dbxtract) 之前创建一个空数据库。例如，以下命令可以创建一个名为 *mydata.db* 的空数据库：

```
dbinit c:\remote\mydata.db
```

运行以下命令以从位于 *c:\consolidateddata.db* 中的统一数据库抽取新的远程数据库。新数据库可用于名为 field_user 的远程用户，并且是在 *c:\remote\mydata.db* 中创建的：

```
dbxtract -c "UID=DBA;PWD=sql;DBF=c:\consolidateddata.db"
-an c:\remote\mydata.db field_user
```

具有适当模式、远程用户、发布、预订和触发器的新远程数据库 *mydata.db* 将会创建。缺省情况下，会将统一数据库中的数据抽取到远程数据库，并启动预订。但是，抽取实用程序 (dbxtract) 不会启动消息代理，所以没有消息被交换。请参见“了解消息代理 (dbremote)”一节第 86 页。

另请参见

- “将远程数据库抽取到重装文件”一节第 81 页

将远程数据库抽取到重装文件

有关将远程数据库抽取到重装文件及其替代方法（自动抽取远程数据库）的信息，请参见“抽取远程数据库”一节第 79 页。

在大部分部署情况下，需要自定义对远程数据库的抽取和创建。可以通过选择将数据库抽取到命令文件和一系列文本文件来创建自定义抽取。然后，根据需要编辑这些文件。

将数据库抽取到文件中时，确定是否创建以下文件：

- **名为 *reload.sql* 的 SQL 命令文件，该文件包含建立远程数据库模式所必需的语句。** 请参见 -n 选项“抽取实用程序 (dbxtract)”一节第 147 页。

  例如，可运行以下命令：

  ```
  dbxtract -c "UID=DBA;PWD=sql;DBF=c:\cons\cons.db" -n "c:\remote
  \reload.sql" UserName
  ```

- **一系列数据文件，其中每个文件都包含数据库表的内容。** 一系列数据文件，其中每个文件都包含数据库表的内容。创建名为 *extract* 的新目录以包括这些数据文件。可以使用这些文件将数据装载到现有远程数据库中。请参见 -d 选项“抽取实用程序 (dbxtract)”一节第 147 页。

  例如，可运行以下命令：

  ```
  dbxtract -c "UID=DBA;PWD=sql;DBF=c:\cons\cons.db" -d "c:\remote1\" UserName
  ```

- ***reload.sql* 文件和数据文件。** 创建名为 *extract* 的新目录以包括这些数据文件。*reload.sql* 文件包含装载数据文件的说明。例如，可运行以下命令：

  ```
  dbxtract -c "UID=DBA;PWD=sql;DBF=c:cons\cons.db" "c:\remote1\reload.sql"
  UserName
  ```

reload.sql 文件

reload.sql 文件包含用于建立数据库模式的 SQL 语句，而这些语句中包括用来创建以下内容的命令：

- 发布者、远程用户和统一用户
- 发布和预订
- 消息类型
- 表
- 视图
- 触发器
- 过程

可能需要编辑 reload.sql
抽取实用程序 (dbxtract) 可用于在准备远程数据库的过程中提供帮助，但并不适用于所有情况下的黑箱解决方案。在创建远程数据库时应根据需要编辑 *reload.sql* 命令文件。请参见“编辑 reload.sql 文件”一节第 82 页。

通过以下过程来使用 reload.sql 文件创建远程数据库。

♦ **从 reload.sql 文件（命令行）创建远程数据库**

1. 使用抽取实用程序 (dbxtract) 将数据库模式和数据抽取到文件中。例如，可运行以下命令：

   ```
   dbxtract -c "UID=DBA;PWD=sql;DBF=c:cons\cons.db" "c:\remote\reload.sql"
   UserName
   ```

   缺省情况下，自动启动指定远程用户的预订。

2. 在需要时，可以编辑 *reload.sql*。请参见“编辑 reload.sql 文件”一节第 82 页。

3. 创建空 SQL Anywhere 数据库。

   例如，可运行以下命令：

   ```
   dbinit c:\rem1\rem1.db
   ```

4. 通过 Interactive SQL 连接到数据库，并运行 *reload.sql* 命令文件。

   例如，可运行以下命令：

   ```
   READ remote\reload.sql
   ```

   具有适当模式、远程用户、发布、预订和触发器的新远程数据库 *rem1.db* 将会创建。但是，抽取实用程序 (dbxtract) 不会启动消息代理，所以没有消息被交换。请参见“了解消息代理 (dbremote)”一节第 86 页。

另请参见

- “自动抽取远程数据库”一节第 79 页

编辑 reload.sql 文件

在创建远程数据库时应根据需要编辑 *reload.sql* 命令文件。例如，在以下情况下必须编辑 *reload.sql* 文件：

将未复制的表添加到远程数据库

如果远程数据库中的表不参与复制，那么就不会出现在统一数据库中。抽取实用程序 (dbxtract) 和 [**抽取数据库向导**] 无法从统一数据库抽取未复制的表。

抽取数据库之后，应编辑 *reload.sql* 以添加此类表。

抽取过程、触发器和视图

缺省情况下，抽取实用程序 (dbxtract) 和 [**抽取数据库向导**] 会从数据库抽取所有存储过程、触发器和视图。远程站点可能需要这些视图和过程中的一部分，而不需要其它部分。例如，一个过程可能只引用远程站点所不包含的那部分数据库。

抽取数据库之后，应编辑 *reload.sql* 以删除不必要的过程、触发器和视图。

在多层系统中使用抽取实用程序 (dbxtract)

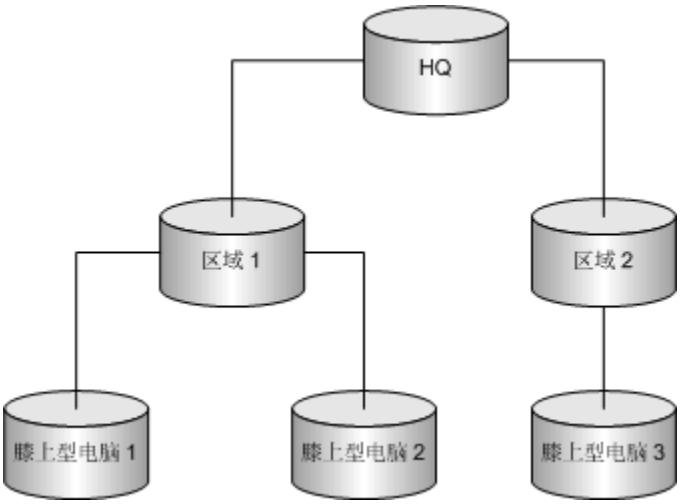
请参见“为多层系统抽取数据库”一节第 83 页。

另请参见

- “创建多个远程数据库”一节第 84 页
- “将远程数据库抽取到重装文件”一节第 81 页

为多层系统抽取数据库

若要了解抽取实用程序 (dbxtract) 和 [**抽取数据库向导**] 在多层结构中的作用，请考虑三层 SQL Remote 系统。此系统在下图中进行了说明。



♦ **为三层系统创建远程数据库**

1. 在第一层（即统一数据库 HQ）上使用抽取实用程序 (dbxtract)，以创建第二层数据库“区域 1”和“区域 2”。
2. 在第二层数据库“区域 1”和“区域 2”上使用抽取实用程序 (dbxtract)，以为用户“膝上型电脑 1”、“膝上型电脑 2”和“膝上型电脑 3”创建第三层数据库。第二层数据库是第一层数据库 HQ 的远程数据库，也是第三层数据库“膝上型电脑 1”、“膝上型电脑 2”和“膝上型电脑 3”的统一数据库。

在多层系统中重新抽取数据库

如果必须从第一层统一数据库重新抽取第二层数据库的*模式*，则抽取实用程序 (dbxtract) 会删除远程用户（膝上型电脑 1、膝上型电脑 2 和膝上型电脑 3）及其预订和权限。因此，必须手工重新创建这些第三层用户及其预订。

如果只需从第一层统一数据库重新抽取第二层数据库的*数据*，则抽取实用程序 (dbxtract) 不会影响远程用户。请参见 -d 选项“抽取实用程序 (dbxtract)”一节第 147 页。

完全限定的发布定义

完全限定的发布定义包含 WHERE 和 SUBSCRIBE BY 子句。大多数情况下，您不需要为远程数据库抽取完全限定的发布定义，因为远程数据库通常会将所有行复制回统一数据库。

另请参见

- “创建多个远程数据库”一节第 84 页
- “将远程数据库抽取到重装文件”一节第 81 页

创建多个远程数据库

通过以下步骤提高创建多个远程数据库的效率。

♦ 创建多个远程数据库

1. 创建统一数据库的副本，并从统一数据库启动远程用户的预订。例如：
   a. 启动预订，然后立即关闭统一数据库和消息代理（如果它正在运行）。

      必须在创建统一数据库副本的同时启动预订。在复制数据库和启动预订之间发生的任何操作都将丢失，而且可能在远程数据库上导致错误。在统一数据库上启动预订允许将消息封装并发送到预订者，即使预订者数据库尚不存在也会如此。请参见“START SUBSCRIPTION 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》和“启动预订”一节第 134 页。

      若要在单个事务中启动多个预订，请使用 REMOTE RESET 语句。请参见“REMOTE RESET 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。

   b. 复制统一数据库。

      缺省情况下，抽取实用程序 (dbxtract) 和 [**抽取数据库向导**] 都在隔离级别 3 上运行。此隔离级别可以确保抽取的数据库中的数据与数据库服务器上的数据保持一致；但是，它会阻止其他用户使用数据库。建议针对统一数据库的副本抽取远程数据库。

   c. 重新启动统一数据库，如果它正在运行，则重新启动该统一数据库上的消息代理。
2. 从统一数据库的副本抽取远程数据库模式。因为该数据库是一个副本，所以没有锁定和并发问题；不过，对于大量的远程数据库，这一过程会花费一段时间。

   抽取远程数据库模式时，请选择以下选项：

   a. 仅抽取远程数据库的模式。

      缺省情况下，抽取实用程序 (dbxtract) 和 [**抽取数据库向导**] 一次抽取一个数据库，包括每个用户的模式和数据。但在大部分部署情况下，远程数据库使用相同的模式但使用不同的数据。使用抽取实用程序 (dbxtract) 或 [**抽取数据库向导**] 为每个用户同时抽取模式和数据会导致重复抽取同一模式。请参见 -n 选项“抽取实用程序 (dbxtract)”一节第 147 页。

   b. 按主键对数据排序。

      缺省情况下，每个表中的数据都按主键排序。如果数据是按主键进行排序的，则将数据装载至远程数据库将会更快。请参见 -u 选项“抽取实用程序 (dbxtract)”一节第 147 页。
3. 使用 *reload.sql* 文件创建空的远程数据库。复制此数据库文件以创建所需数量的远程数据库。请参见“将远程数据库抽取到重装文件”一节第 81 页。
4. 针对每个远程数据库，定义特定于每个远程用户的 SQL Remote 定义。请参见“用户权限”一节第 21 页。

5. 针对每个远程用户，从统一数据库中仅抽取其相应数据。请参见 -d 选项“抽取实用程序 (dbxtract)”一节第 147 页。
6. 将每个远程用户的数据都装载到相应的远程数据库中。

   每个远程数据库在创建时，对于正在使用的统一数据库来说已经过期。

   但是，当运行消息代理 (dbremote) 时，每个用户都可以接收并应用从该活动的统一数据库中发出的消息，这样就可使这些数据库处于最新状态。请参见“了解消息代理 (dbremote)”一节第 86 页。

另请参见

- “为多层系统抽取数据库”一节第 83 页
- “编辑 reload.sql 文件”一节第 82 页

了解消息代理 (dbremote)

消息代理 (dbremote) 是 SQL Remote 复制中的重要组件。它必须在系统的每个数据库中进行安装和运行。消息代理 (dbremote) 负责发送和接收消息。

它执行以下功能：

- **发送消息时消息代理 (dbremote) 的任务**
  - 扫描各个发布者数据库的事务日志，并将事务日志条目转换为要发送给预订者的消息。
  - 将消息发送给预订者。
  - 当接收到要重新发送消息的请求时，消息代理 (dbremote) 会将消息重新发送到发出请求的数据库。
  - 在系统表中维护消息信息，并管理保证消息传送系统。

  请参见“发送消息任务”一节第 96 页。
- **接收消息时消息代理 (dbremote) 的任务**
  - 处理进来的消息，并以适当的顺序将其应用到数据库中。
  - 请求重新发送丢失的消息。
  - 在系统表中维护消息信息，并管理保证消息传送系统。

  请参见“接收消息任务”一节第 91 页。

连接

消息代理 (dbremote) 使用多个数据库服务器连接。它们是：

- **一个全局连接** 此连接在消息代理 (dbremote) 运行时一直处于活动状态。
- **一个用于扫描事务日志的连接** 此连接仅在扫描阶段处于活动状态。
- **一个用于从事务日志扫描线程执行命令的连接** 此连接仅在扫描阶段处于活动状态。
- **一个用于处理同步预订请求的连接** 此连接仅在发送阶段处于活动状态。
- **一个用于每个工作线程的连接** 这些连接仅在接收阶段处于活动状态。

选择消息代理 (dbremote) 模式

可以使用以下两种模式之一运行消息代理 (dbremote)：

- **连续模式** 在连续模式下，消息代理 (dbremote) 会按由每个远程用户的发送频率属性指定的时间定期发送消息。不发送消息时，消息代理会在消息到达时接收消息。

  连续模式可用于统一数据库（在这些数据库中，随时可能有消息进来或发出），以分散工作量并确保及时复制。

  请参见“在连续模式下运行消息代理 (dbremote)”一节第 87 页。

- **批处理模式** 在批处理模式下，消息代理 (dbremote) 接收并处理进来的消息，扫描一次事务日志，创建并发送外发消息，然后停止。

  批处理模式可用于偶尔进行连接的远程数据库，在这些数据库中，建立连接后（例如，当远程数据库通过拨号连接到主网络后）才能与统一数据库交换消息。

  请参见“在批处理模式下运行消息代理 (dbremote)”一节第 89 页。

消息代理 (dbremote) 的要求

SQL Remote 非常灵活。在系统中，您可以在两种模式下于多台设备及多个操作系统上运行消息代理 (dbremote)。但是，SQL Remote 具有以下要求：

- **需要 REMOTE DBA 权限或 DBA 权限** 消息代理 (dbremote) 必须由具有 REMOTE DBA 权限或 DBA 权限的用户来运行。请参见“授予 REMOTE DBA 权限”一节第 30 页。
- **对于系统中的每个消息代理 (dbremote)，最大消息长度必须相同。** 此长度会受到操作系统内存分配限制的制约。如果收到的消息长度超过该限制，将把它们作为损坏消息加以删除。缺省值是 50000 个字节。可以使用消息代理 (dbremote) 的 -l 选项对该长度进行配置。请参见“消息代理 (dbremote)”一节第 140 页。

在连续模式下运行消息代理 (dbremote)

统一数据库通常在连续模式下运行。有关连续模式及其替代模式（批处理模式）的信息，请参见“了解消息代理 (dbremote)”一节第 86 页。

♦ 在连续模式下运行消息代理 (dbremote)

1. 确保每个具有 REMOTE 权限的用户都指定了 SEND AT 或 SEND EVERY 频率。

   在连续模式下，消息代理 (dbremote) 会按在每个远程用户的属性中由 SEND AT 或 SEND EVERY 频率指定的时间来发送消息。请参见“设置发送频率”一节第 88 页。

2. *不使用* -b 选项启动消息代理 (dbremote)。

   在 Windows 上，消息代理 (dbremote) 名为 *dbremote.exe*。在 Unix 上，它的名称是 dbremote。在 Mac OS X 上，也可以使用 SyncConsole 启动消息代理 (dbremote)。请参见“在 Mac OS X 上运行消息代理 (dbremote)”一节第 89 页和“在 Unix 上运行消息代理 (dbremote)”一节第 90 页。

   例如，以下语句将在连续模式下于名为 *c:\mydata.db* 的数据库文件中运行 dbremote，并使用用户名 ManagerSteve 和口令 sql 进行连接：

   ```
   dbremote -c "UID=ManagerSteve;PWD=sql;DBF=c:\mydata.db" -l 40000
   ```

   用户名 ManagerSteve 必须具有 REMOTE DBA 权限或 DBA 权限。对于系统中的所有数据库，由 -l 选项定义的最大消息长度必须相同。请参见“消息代理 (dbremote) 的要求”一节第 87 页。

   有关可以指定的 dbremote 选项的完整列表，请参见“消息代理 (dbremote)”一节第 140 页。

在连续模式下将消息代理 (dbremote) 作为服务运行

当在连续模式下运行消息代理 (dbremote) 时，只要数据库服务器在运行，就可以选择使消息代理 (dbremote) 保持运行状态。通过将消息代理 (dbremote) 作为 Windows 服务来运行，可以实现这一目的。可将服务配置为即使当前用户已注销也可以保持运行状态，并且在操作系统启动时启动。

有关将程序作为服务运行的完整说明，请参见“用于 Windows 的服务实用程序 (dbsvc)”一节《SQL Anywhere 服务器 - 数据库管理》。

设置发送频率

要在连续模式下运行消息代理 (dbremote)（例如在统一数据库上），则必须确保每个远程用户都已指定发送频率。在连续模式下，消息代理 (dbremote) 会按通过 SEND AT 或 SEND EVERY 属性指定的时间来发送消息。

消息代理 (dbremote) 支持以下发送频率值：

- **SEND EVERY** 指定在发送消息之间要等待的时间长度。

  将消息发送给任何设置了 SEND EVERY 的用户时，会将消息发送给具有相同发送频率的所有用户。例如，对于每十二个小时接收一次更新的所有远程用户，系统会将更新同时发送给他们，而不是分开发送。这样便会减少需要对 SQL Anywhere 事务日志进行处理的次数。应尽量少用不同的频率。

  可以按照 ***HH*:*MM*:*SS*** 格式，以小时、分钟和秒指定发送频率。

- **SEND AT** 指定在一天的什么时间发送消息。

  每天在指定时间发送更新。应尽量只用几个不同的发送时间，而不是分别指定不同的发送时间。应选择数据库不忙的时间。

- **缺省设置（不使用 SEND 子句）** 如果所有用户都未指定 SEND AT 或 SEND EVERY 子句，则消息代理 (dbremote) 在批处理模式下运行，并且在每次运行时发送消息，然后停止。请参见“在批处理模式下运行消息代理 (dbremote)”一节第 89 页。

发送消息过于频繁

如果频繁地发送消息，则发送短消息的可能性会更大。如果以较低的频率发送消息，那么一条消息中就可以包含更多指令。如果大量短消息对您的消息系统来说是个问题，则应避免使用过短的发送频率周期。

♦ 设置发送频率 (Sybase Central)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 在左窗格中，选择 [**SQL Remote 用户**] 目录。
3. 右击用户，然后选择 [**属性**]。
4. 单击 [**SQL Remote**] 选项卡。
5. 选择 [**发送时间间隔**] 或 [**每天发送时间**]，并指定一个时间。

另请参见

- “GRANT REMOTE 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》

在批处理模式下运行消息代理 (dbremote)

通过以下过程在批处理模式下运行 SQL Remote。有关批处理模式及其替代模式（连续模式）的信息，请参见“了解消息代理 (dbremote)”一节第 86 页。

♦ 在批处理模式下运行消息代理 (dbremote)

1. 确保至少有一个远程用户的远程属性中既未指定 SEND AT 选项也未指定 SEND EVERY 选项。

   如果**所有**远程用户都已定义 SEND AT 或 SEND EVERY 子句，并且您希望消息代理 (dbremote) 在发送和接收消息之后关闭，则必须使用 -b 选项来启动消息代理 (dbremote)。

2. 启动消息代理 (dbremote)。

   在 Windows 上，消息代理 (dbremote) 名为 *dbremote.exe*。在 Unix 上，它的名称是 dbremote。在 Mac OS X 上，也可以使用 **SyncConsole** 启动消息代理 (dbremote)。请参见“在 Mac OS X 上运行消息代理 (dbremote)”一节第 89 页和“在 Unix 上运行消息代理 (dbremote)”一节第 90 页。

   例如，以下语句将在名为 *c:\mydata.db* 的数据库文件中且在批处理模式下运行 dbremote，并使用用户名 ManagerSteve 和口令 sql 进行连接：

   ```
   dbremote -c "UID=ManagerSteve;PWD=sql;DBF=c:\mydata.db"
   ```

   消息代理 (dbremote) 接收并处理进来的消息，扫描一次事务日志，创建并发送外发消息，然后停止。

   用户名 ManagerSteve 必须具有 REMOTE DBA 权限或 DBA 权限。对于系统中的所有数据库，由 -l 选项定义的最大消息长度必须相同。请参见“消息代理 (dbremote) 的要求”一节第 87 页。

另请参见

- “消息代理 (dbremote)”一节第 140 页

在 Mac OS X 上运行消息代理 (dbremote)

SQL Anywhere 包括一个名为 SyncConsole 的应用程序，该程序可以用来在 Mac OS X 上启动消息代理 (dbremote)。您也可以使用 dbremote 实用程序在 Mac OS X 上启动消息代理。

♦ 启动 SyncConsole

1. 在 Finder 中，转到 */Applications/SQLAnywhere11*。
2. 双击 [**SyncConsole**]。
3. 选择 [**文件**] » [**新建**] » [**SQL Remote**]。

将出现客户端选项窗口。

4. 为消息代理 (dbremote) 指定连接信息。

   例如，以下连接参数使用 SQL Anywhere 示例数据库的一个 ODBC 数据源：

   ```
   DSN="SQL Anywhere 11 Demo"
   ```

   该用户名必须具有 REMOTE DBA 权限或 DBA 权限。由 -l 选项定义的消息长度在系统的所有数据库中必须相同。请参见“消息代理 (dbremote) 的要求”一节第 87 页。

   有关可以在 [**选项**] 字段中指定的 dbremote 选项的完整列表，请参见“消息代理 (dbremote)”一节第 140 页。

在 Unix 上运行消息代理 (dbremote)

在 Unix 平台上，可以通过提供 -ud 选项将消息代理 (dbremote) 作为守护程序来运行。请参见 -ud 选项“消息代理 (dbremote)”一节第 140 页。

该用户名必须具有 REMOTE DBA 权限或 DBA 权限。由 -l 选项定义的最大消息长度在系统的所有数据库中必须相同。请参见“消息代理 (dbremote) 的要求”一节第 87 页。

有关可以指定的 dbremote 选项的完整列表，请参见“消息代理 (dbremote)”一节第 140 页。

提高 SQL Remote 的性能

每次插入、删除或更新表中的行时，都会为那些已预订该行的用户创建消息。此外，更新可能引起预订表达式更改，所以更新语句将分别作为删除语句、更新语句和插入语句发送给某些相应的预订者。

数据库服务器和消息代理 (dbremote) 共同确定某一接收者将获得哪些操作。

数据库服务器

数据库服务器处理发布。请参见“数据库服务器处理发布”一节第 35 页。

消息代理 (dbremote)

消息代理 (dbremote) 处理预订。

消息代理 (dbremote) 从事务日志中读取已计算的预订表达式或预订列条目，并为该发布的每个预订者查找与预订值匹配的前后值。通过这种方式，消息代理 (dbremote) 可向每个预订者发送正确的操作。

虽然大量的预订者对数据库服务器性能没有任何影响，但他们可能会影响消息代理 (dbremote) 的性能。查找与大量预订值匹配的预订值的工作以及发送消息的工作都可能要求很高。

接收消息任务

消息代理 (dbremote) 在接收消息时会执行以下任务：

- **轮询进来的消息** 要检查是否存在已到达数据库的新消息，消息代理 (dbremote) 会轮询新消息。请参见“调整轮询间隔以检查新消息”一节第 92 页。
- **读取消息** 消息到达后，会在应用前由消息代理 (dbremote) 读取并存储在高速缓存中。请参见“通过高速缓存消息调整吞吐量”一节第 93 页。

  如果某个消息丢失，且消息代理 (dbremote) 正在连续模式下运行，则消息代理 (dbremote) 会等待该消息到达后继续轮询。消息代理 (dbremote) 等待的轮询数被称为其**等待时间**，并且由 -rp 选项指定。

  - 如果丢失的消息在消息代理 (dbremote) 的等待时间到期之前到达，则该丢失的消息将按正确的顺序添加到高速缓存。
  - 如果丢失的消息未到达而消息代理 (dbremote) 的等待时间已到期，则消息代理 (dbremote) 会发送请求，要求从发布者数据库重新发送该消息。

  消息会继续被读取并添加至高速缓存，直到超出高速缓存的使用量。当超过通过 -m 选项指定的高速缓存使用量时，消息将被删除。

  请参见“调整要重新发送消息的请求”一节第 94 页。
- **应用消息** 消息代理 (dbremote) 按正确的顺序将消息应用到预订者数据库。请参见“调整工作线程数”一节第 95 页。

- **等待消息已应用到预订者数据库的确认** 在预订的数据库中收到并应用了该消息后，将把确认信息返回给发布者。发布者消息代理 (dbremote) 在接收到确认后，将在系统表中跟踪该确认。请参见“了解保证消息传送系统”一节第 99 页。

在接收消息时提高性能

SQL Remote 系统总吞吐量的主要瓶颈一般是，从众多远程数据库接收消息并将这些消息应用到数据库。要减少此延迟时间，可在连续模式下运行消息代理 (dbremote) 时，调整以下变量：

- 消息代理 (dbremote) 检查进来的消息的频率。请参见“调整轮询间隔以检查新消息”一节第 92 页。
- 消息代理 (dbremote) 用于保存要发送的消息的内存量。请参见“通过高速缓存消息调整吞吐量”一节第 93 页。
- 消息代理 (dbremote) 在请求重新发送未按顺序到达的消息之前等待该消息到达的时间。请参见“调整要重新发送消息的请求”一节第 94 页。
- 用于处理接收的消息的工作线程数。请参见“调整工作线程数”一节第 95 页。

调整轮询间隔以检查新消息

要检查是否存在已到达数据库的新消息，消息代理 (dbremote) 会轮询新消息。缺省的轮询间隔（从一次轮询结束到下一次轮询开始）为 1 分钟。可以使用 -rd 选项配置轮询间隔，但通常缺省值便足够。

减小轮询间隔

通过使用以秒为单位的值可以更频繁地轮询。例如，以下命令每三十秒轮询一次：

```
dbremote -c "DSN=SQL Anywhere 11 Demo" -rd 30s
```

一般来说，除非有特殊原因需要对消息作出快速响应，否则不要使用非常短的轮询间隔。将间隔设置得非常短可能会对系统的整体吞吐量产生不利影响，因为：

- 当队列中没有消息时进行轮询会浪费资源。例如，如果正在使用电子邮件，则每次轮询邮件服务器时都会增加消息系统的负荷。轮询太频繁可能会影响消息系统，并且不会带来任何益处。
- 重新发送请求会使您的系统过载。调整轮询间隔时，还应调整消息代理 (dbremote) 的等待时间。此等待时间是消息代理 (dbremote) 在请求重新发送未按顺序到达的消息之前等待该消息到达所经历的轮询数。请参见“调整要重新发送消息的请求”一节第 94 页。

增大轮询间隔

可以较低的频率轮询，如以下命令所示（每五分钟轮询一次）：

```
dbremote -c "DSN=SQL Anywhere 11 Demo" -rd 5
```

设置较长的轮询间隔可以提高系统整体的消息吞吐量，但会增加应用各条消息的时间。例如，如果检查进来的消息的轮询周期与消息到达的频率相比过长，则结果可能是消息搁置在队列中，等待接受处理。

另请参见

- “在接收消息时提高性能”一节第 92 页
- “通过高速缓存消息调整吞吐量”一节第 93 页
- “调整要重新发送消息的请求”一节第 94 页
- “调整工作线程数”一节第 95 页
- “在发送消息时提高性能”一节第 97 页

通过高速缓存消息调整吞吐量

消息到达后，消息代理 (dbremote) 会在其被应用之前读取并将其存储在高速缓存中。将消息存入高速缓存可以防止：

- 从消息系统中重新读取顺序错乱的消息。在大型系统中，重新读取消息会降低性能。当通过 WAN（例如，通过调制解调器连接的远程访问服务或 POP3）读取消息时，将消息存入高速缓存非常有用。
- 读取消息（单线程任务）的数据库工作线程之间发生争用，因为消息的内容已被高速缓存。

消息的高速缓存机制

发生以下情况之一时，消息代理 (dbremote) 会在消息被应用之前将其存储在内存中：

- 事务太大，以至于需要由多个部分组成的消息。
- 消息未按正确顺序到达。

指定消息高速缓存的大小

使用消息代理 (dbremote) 的 -m 选项指定消息高速缓存的大小。-m 选项指定消息代理 (dbremote) 用于保存消息的最大内存量。允许的大小可以指定为 *n*（以字节为单位）、*n*K 或 *n*M。缺省值为 2048K (2M)。当超过指定的高速缓存使用量时，消息将被删除。

如果您拥有一个统一数据库和大量的远程数据库，则 -m 选项非常有用。请参见“消息代理 (dbremote)”一节第 140 页中的 -m 选项。

示例

以下命令使用 12 MB 内存作为消息高速缓存来启动消息代理 (dbremote)：

```
dbremote -c "DSN=SQL Anywhere 11 Demo" -m 12M
```

另请参见

- “在接收消息时提高性能”一节第 92 页
- “调整轮询间隔以检查新消息”一节第 92 页
- “调整要重新发送消息的请求”一节第 94 页
- “调整工作线程数”一节第 95 页
- “在发送消息时提高性能”一节第 97 页

调整要重新发送消息的请求

当队列中某一消息丢失时，消息代理 (dbremote) 会在请求重新发送该丢失的消息之前等待指定数量的轮询。消息代理 (dbremote) 等待的轮询数称为其等待时间。缺省情况下，消息代理 (dbremote) 的等待时间为 1。

如果消息代理 (dbremote) 的等待时间为 1，并且应接收到消息 6，但却接收到消息 7，则它不会执行任何操作。相反，消息代理 (dbremote) 会等待下一次轮询的结果。如果在下一次轮询之后仍未接收到消息 6，则消息代理 (dbremote) 会发出请求，要求重新发送消息 6。

增加重新发送的等待时间

假设轮询间隔非常短，并且消息系统不会保留消息到达的顺序。顺序错乱的消息在两次或三次轮询完成以后到达的情况可能很常见。在此示例中，建议使用 -rp 选项来增加消息代理 (dbremote) 的等待时间，以免发送大量不必要的重新发送请求。-rp 选项通常与设置轮询间隔的 -rd 选项一起使用。请参见“调整轮询间隔以检查新消息”一节第 92 页。

示例

有两个远程用户（名为 user1 和 user2），都以 30 秒的轮询间隔和 3 次轮询的等待时间来运行消息代理 (dbremote)。例如，他们使用以下命令运行消息代理 (dbremote)：

```
dbremote -c "DSN=SQL Anywhere 11 Demo"  -rd 30s -rp 3
```

在以下一系列操作中，消息被标记为 *userX.n*，其中 X 为用户名，*n* 为消息号。例如，user1.5 是来自 user1 的第 5 条消息。消息代理 (dbremote) 预期两个用户的消息都从编号 1 开始。

在 0 秒时：

1. 消息代理 (dbremote) 读取 user1.1、user2.4
2. 消息代理 (dbremote) 应用 user1.1
3. 消息代理 (dbremote) 的等待时间现在为 user1：未知，user2：3，因为来自 user2 的失序消息已到达

在 30 秒时：

1. 消息代理 (dbremote) 读取：无新消息
2. 消息代理 (dbremote) 应用：无
3. 消息代理 (dbremote) 的等待时间现在为 user1：未知，user2： 2

在 60 秒时：

1. 消息代理 (dbremote) 读取：user1.3
2. 消息代理 (dbremote) 应用：无新消息
3. 消息代理 (dbremote) 的等待时间：user1：3，user2： 1

在 90 秒时：

1. 消息代理 (dbremote) 读取：user1.4
2. 消息代理 (dbremote) 应用：无
3. 消息代理 (dbremote) 的等待时间为 user1：3，user2：0
4. 消息代理 (dbremote) 向 user2 发出重发请求

当用户接收到新消息时，消息代理 (dbremote) 的等待时间将被重置，即使该消息不是预期的消息。

在 120 秒时：

1. 消息代理 (dbremote) 读取：user1.2 和 user2.2
2. 消息代理 (dbremote) 应用 user1.2、user1.3、user1.4 和 user2.2
3. 消息代理 (dbremote) 的等待时间为 user1：未知，user2：N/A

另请参见

- “在接收消息时提高性能”一节第 92 页
- “调整轮询间隔以检查新消息”一节第 92 页
- “通过高速缓存消息调整吞吐量”一节第 93 页
- “调整工作线程数”一节第 95 页
- “在发送消息时提高性能”一节第 97 页

调整工作线程数

以下步骤介绍消息代理 (dbremote) 如何应用进来的消息：

1. 它读取消息。消息将被读取且标头信息将被检查（以确定正确的应用顺序）。从消息系统读取消息的操作是单线程的。
2. 它应用消息。将读取的消息传递给数据库工作线程以进行应用。

   在远程数据库中，通常以串行方式应用消息。在多层系统中，远程数据库也可以作为其它远程数据库的统一数据库。在此类远程数据库中，消息的应用方式与统一数据库中一样。

   在统一数据库中，缺省设置为以串行方式应用消息。可以使用附加的数据库工作线程以并行方式应用远程用户发来的消息。请参见“消息代理 (dbremote)”一节第 140 页中的 -w 选项。

   当在统一数据库中使用数据库工作线程时：

   - 不同远程用户发来的消息将以并行方式进行应用。
   - 来自一个远程用户的消息将以串行方式应用。

     例如，来自同一远程用户的十条消息将由一个工作线程按正确的顺序应用。

使用数据库工作线程的优点

在统一数据库中使用数据库工作线程可提高吞吐量，方法是以并行方式（而不是串行方式）应用消息。如果数据库服务器位于具有条纹化驱动器阵列的系统上，则性能优势将非常显著。

使用数据库工作线程的缺点

如果在统一数据库中使用数据库工作线程导致用户之间的大量锁定，则会降低吞吐量。

死锁将通过稍后重新应用已回退的事务来进行处理。

♦ 设置数据库工作线程的数目

- 在统一数据库中，使用 -w 选项设置数据库工作线程的数目。

  例如，以下命令可将工作线程数设置为 5：

  ```
  dbremote -c "DSN=SQL Anywhere 11 Demo"  -w 5
  ```

另请参见

- “在连续模式下运行消息代理 (dbremote)”一节第 87 页
- “在接收消息时提高性能”一节第 92 页
- “调整轮询间隔以检查新消息”一节第 92 页
- “通过高速缓存消息调整吞吐量”一节第 93 页
- “调整要重新发送消息的请求”一节第 94 页
- “在发送消息时提高性能”一节第 97 页

发送消息任务

消息代理 (dbremote) 执行以下任务以发送消息：

- **扫描发布者事务日志** 消息代理 (dbremote) 扫描发布者数据库的事务日志，并为预订者将事务日志条目转换为消息。由 -l 选项定义的最大消息长度在系统的所有数据库中必须相同。

  对于大事务，消息代理 (dbremote) 会创建由多个部分组成的消息。所有这些消息都包含一个序列号，用来跟踪它们在事务中的位置。预订者数据库中的消息代理 (dbremote) 使用该序列号，以确保按正确顺序应用消息，并且不会丢失消息。

- **将消息发送到远程数据库** 消息代理 (dbremote) 会按由每个远程用户的发送频率属性指定的时间发送消息。请参见“调整发送延迟”一节第 97 页。

  如果消息代理 (dbremote) 的高速缓存超过设置值，则消息代理 (dbremote) 会提前发送消息。消息代理 (dbremote) 将其消息存储在高速缓存中。当所用的高速缓存超过指定值时，将发送消息。请参见“通过高速缓存消息调整吞吐量”一节第 97 页。

- **处理远程数据库的重发请求** 当用户请求重新发送消息时，发布者数据库中的消息代理 (dbremote) 会中断常规消息发送过程，以处理重发请求。

  可以控制使用 -ru 选项处理这些重发请求时执行的紧急程度。请参见“调整处理重发请求的速度”一节第 98 页。

- **将确认发送到发布者数据库** 在预订的数据库中收到并应用了消息后，会将确认返回给发布者。请参见“了解保证消息传送系统”一节第 99 页。

在发送消息时提高性能

发送消息的主要性能问题是在一个站点上输入数据的时间与该数据出现在其它站点上的时间之间的周转时间。要减小此延迟时间，可在通过消息代理 (dbremote) 发送消息时调整以下变量：

- 向远程数据库发送消息的频率。请参见“调整发送延迟”一节第 97 页。
- 消息的大小。请参见“在接收消息时提高性能”一节第 92 页。
- 处理重发请求的速度。请参见“调整处理重发请求的速度”一节第 98 页。

调整发送延迟

如果要创建要发送的消息，则消息代理 (dbremote) 将为新数据从事务日志进行轮询。发送延迟是对于要发送的更多事务日志数据而进行的各轮询之间的间隔时间。缺省的轮询间隔（从一次轮询结束到下一次轮询开始）为 1 分钟。可以使用 -sd 选项配置发送延迟，但通常缺省值便足够。发送延迟应小于或等于远程用户的发送频率。

减小发送延迟

通过使用以秒为单位的值可以更频繁地轮询。例如，以下命令每三十秒轮询一次：

```
dbremote -c "DSN=SQL Anywhere 11 Demo"  -sd 30s ...
```

增大发送延迟

可以较低的频率轮询，如以下命令所示（每 60 分钟轮询一次）：

```
dbremote -c "DSN=SQL Anywhere 11 Demo"  -sd 60
```

设置较大的间隔通常需要消息代理 (dbremote) 在发送消息之前处理大部分的消息创建工作。通常会偏好设置较小的间隔，因为这样可以分散消息创建工作。

请参见“消息代理 (dbremote)”一节第 140 页。

另请参见

- “通过高速缓存消息调整吞吐量”一节第 97 页
- “调整处理重发请求的速度”一节第 98 页

通过高速缓存消息调整吞吐量

消息代理 (dbremote) 将要发送的消息高速缓存在可配置的内存区内。

当所有远程数据库接收的是所复制操作的唯一子集时，将同时为每一远程数据库分别生成不同的消息。为接收相同操作的一组远程用户只生成一个消息。在以下情况下将发送消息：

- 达到发送频率。
- 当所用的高速缓存超过 -m 值时。
- 当消息大小达到其最大值（由 -l 选项指定）时。

指定消息高速缓存的大小

在消息代理 (dbremote) 命令上使用 -m 选项指定消息高速缓存的大小。

-m 选项指定消息代理 (dbremote) 用于创建消息的最大内存量。允许的大小可以指定为 *n*（以字节为单位）、*n*K 或 *n*M。缺省值为 2048K (2M)。

如果您拥有一个统一数据库和大量的远程数据库，则 -m 选项非常有用。请参见 -m 选项“消息代理 (dbremote)”一节第 140 页。

示例

以下命令使用 12 MB 内存作为消息高速缓存来启动消息代理 (dbremote)：

```
dbremote -c "DSN=SQL Anywhere 11 Demo" -m 12M
```

另请参见

- “调整发送延迟”一节第 97 页
- “调整处理重发请求的速度”一节第 98 页

调整处理重发请求的速度

因为重新发送消息会中断常规的消息发送过程，所以消息代理 (dbremote) 延迟处理重发请求。缺省情况下，消息代理 (dbremote) 等待的时间等于请求重新发送的远程用户的发送频率的一半。

要重新发送消息，消息代理 (dbremote) 将执行以下任务：

- 它停止扫描事务日志和构建新消息。
- 它删除当前存储在高速缓存中等待发送的消息。消息代理 (dbremote) 在读取事务日志和构建这些消息的过程中所做的所有工作都将丢失。
- 它从重发请求所请求的偏移中重新读取事务日志。消息代理 (dbremote) 构建消息并将其存储在高速缓存中。
- 它一直等到达到下一个发送频率时间，然后发送消息。

必须平衡好发送消息重发请求的紧急程度与处理常规消息的优先级。

-ru 选项控制重发请求的紧急程度。要延迟处理重发请求直到接收到更多的重发请求，请将此选项设置为较长的时间。例如，以下命令将等待一小时，然后再处理重发请求：

```
dbremote -c "DSN=SQL Anywhere 11 Demo" -ru 1h
```

请参见“消息代理 (dbremote)”一节第 140 页。

另请参见

- “调整发送延迟”一节第 97 页
- “通过高速缓存消息调整吞吐量”一节第 97 页
- “调整要重新发送消息的请求”一节第 94 页

了解保证消息传送系统

保证消息传送系统可确保：

- 按正确顺序应用所有复制的操作。请参见“确保按正确顺序应用操作”一节第 100 页。
- 不丢失任何复制的操作。请参见“重新发送丢失或损坏的消息”一节第 101 页。
- 不会将复制的操作应用两次。请参见“确保仅应用消息一次”一节第 102 页。

保证消息传送系统使用以下信息：

- **在 SYSREMOTEUSER 系统表中保留的状态信息** 对于每个预订者，此表中都有与之对应的一行，其中包含向该预订者发送的和该预订者接收的消息的状态信息。例如：
  - 在统一数据库中，对于每个远程用户，SYSREMOTEUSER 系统表都有与之对应的一行。
  - 在每个远程数据库中，SYSREMOTEUSER 系统表都包含一个具有统一数据库信息的单独行。

  SYSREMOTEUSER 系统表由消息代理 (dbremote) 来维护。

  在预订者数据库中，消息代理 (dbremote) 会向发布者数据库发送确认，以确保 SYSREMOTEUSER 系统表在预订所涉及的各方均得到正确维护。

  请参见“ISYSREMOTEUSER 系统表”一节 《SQL Anywhere 服务器 - SQL 参考》。
- **消息标头中的信息** 消息代理 (dbremote) 读取消息的标头信息，并使用此信息更新 SYSREMOTEUSER 系统表。消息的标头包含以下信息：
  - **消息的 resend_count** 跟踪接收数据库丢失消息的次数的计数器。

    在下例中，resend_count 为 1。

    ```
    Current message's header: (1-0000942712-0001119170-0)
    ```
  - **上一条消息中最后一个 COMMIT 的事务日志偏移** 在下例中，上一条消息中最后一次提交操作的事务日志偏移为 0000942712。

    ```
    Previous message's header:(0-0000923357-0000942712-0)
    Current message's header: (0-0000942712-0001119170-0)
    ```
  - **当前消息中上一个 COMMIT 的事务日志偏移** 在下例中，当前消息中上一次提交操作的事务日志偏移为 0001119170：

    ```
    Current message's header: (0-0000942712-0001119170-0)
    ```

    如果事务跨多条消息，则两个事务日志偏移在最后一条消息包含 COMMIT 之前可以相同。

    在下例中，直到第四条消息才发生 COMMIT：

    ```
    (0-0000942712-0000942712-0)
    (0-0000942712-0000942712-1)
    (0-0000942712-0000942712-2)
    (0-0000942712-0001119170-3)
    ```

- **序列号** 当事务跨多条消息时，使用此序列号正确排列消息的顺序。

  序列号零可表示：

  - 如果事务日志偏移不同，则消息不属于包含由多个部分组成的消息的一部分。

    在下例中，消息不属于包含由多个部分组成的消息的一部分：

    ```
    (0-0000923200-0000923357-0)
    (0-0000923357-0000942712-0)
    ```

  - 如果事务日志偏移相同，则消息是由多个部分组成的消息的第一部分。

    在下例中，第一条消息是由多个部分组成的消息的一部分：

    ```
    (0-0000942712-0000942712-0)
    (0-0000942712-0000942712-1)
    (0-0000942712-0000942712-2)
    (0-0000942712-0001119170-3)
    ```

确保按正确顺序应用操作

要确保按正确顺序应用复制的语句，保证消息传送系统使用发布者和预订者数据库的事务日志偏移。每个 COMMIT 指令在事务日志中均由一个明确定义的偏移来标记。事务的顺序可以通过比较其事务日志偏移值来确定。

每条消息都包含以下事务日志偏移：

- 上一条消息中最后一个 COMMIT 指令的事务日志偏移。如果事务跨几条消息，则使用事务内的序列号正确排列消息的顺序。 “确保按正确顺序应用操作”一节第 100 页.
- 消息中最后一个 COMMIT 的事务日志偏移。

消息排序

发送消息时，将按照上一条消息最后一个 COMMIT 指令的偏移对其排序。如果事务跨几条消息，则使用事务内的序列号正确排列消息的顺序。

发送消息

SYSREMOTEUSER 系统表中的 log_sent 列保存发送到预订者的上一条消息的本地事务日志偏移。

下面介绍了当发送消息时如何更新 SYSREMOTEUSER 系统表。

1. 发布者消息代理 (dbremote) 在向预订者发送消息时，还将 log_sent 值设置为发送消息中最后一个 COMMIT 的事务日志偏移值。

   例如，发布者将以下消息发送给 user1。

   ```
   (0-0000923200-0000923357-0)
   ```

   在发布者的 SYSREMOTEUSER 系统表中，发布者为 user1 将 log_sent 值设置为 0000923357。

2. 在预订者数据库中收到并应用消息时，将向发布者发送确认。该确认包含由预订者数据库应用的上一个事务日志偏移。

   例如，消息将确认 user1 已应用了所有更新的且事务日志偏移为 0000923357 的事务。

3. 发布者消息代理 (dbremote) 在接收到确认时，会为 SYSREMOTEUSER 系统表中的用户将 confirm_sent 列设置为确认偏移的值。

   例如，发布者为其 SYSREMOTEUSER 系统表中的 user1 将 confirm_sent 列设置为 0000923357。

log_sent 和 confirm_sent 值都包含发布者事务日志的事务日志偏移。confirm_sent 值的偏移不能晚于 log_sent 值。

接收消息

下面介绍了当接收消息时如何更新 SYSREMOTEUSER 系统表。

1. 预订者数据库中的消息代理 (dbremote) 在接收并应用复制更新时，会以该消息中最后一个 COMMIT 的偏移来更新 SYSREMOTEUSER 系统表中的 log_received 列。

   例如，当预订者接收并应用以下消息时，SYSREMOTEUSER 系统表中的 log_received 值将设置为 0000923357。

   ```
   (0-0000923200-0000923357-0)
   ```

   任何预订者数据库的 log_received 列都包含发布者数据库事务日志中的事务日志偏移。

2. 当接收并应用操作时，预订者消息代理 (dbremote) 将设置其 SYSREMOTEUSER 系统表中的 confirm_received 值，然后向发布者数据库发送确认。

另请参见

- “重新发送丢失或损坏的消息”一节第 101 页
- “确保仅应用消息一次”一节第 102 页

重新发送丢失或损坏的消息

SYSREMOTEUSER 系统表中有两列用来管理重新发送消息：

- **resend_count 列** 跟踪预订者数据库丢失消息的次数的计数器。
- **rereceive_count 列** 跟踪消息代理 (dbremote) 已确定来自发布者用户的消息已丢失的次数的计数器。

当在预订者数据库中按正确顺序接收消息时：

1. 预订者消息代理 (dbremote) 按正确顺序应用消息，并更新其 SYSREMOTEUSER 系统表。
2. 预订者消息代理 (dbremote) 向发布者发送确认消息。
3. 当发布者接收到确认消息时，其消息代理 (dbremote) 会更新其 SYSREMOTEUSER 系统表。

当未按正确顺序接收消息时：

1. 预订者消息代理 (dbremote) 发送重发请求，并增大其 SYSREMOTEUSER 系统表中的 rereceive_count 值。
2. 发布者接收到重发请求时，会为预订者增大 SYSREMOTEUSER 系统表中的 resend_count 值。
3. 在发布者的 SYSREMOTEUSER 系统表中，log_sent 值将设置为 confirm_sent 列中的值。对 log_sent 值进行重置会使操作被重新发送。

另请参见

- “确保按正确顺序应用操作”一节第 100 页
- “确保仅应用消息一次”一节第 102 页

确保仅应用消息一次

预订者消息代理 (dbremote) 会将消息标头中的 resend_count 值与其本地 SYSREMOTEUSER 系统表中的 rereceive_count 值进行比较。如果 resend_count 值小于 rereceive_count，则不会应用消息；消息将被删除。此行为可以确保不会多次应用操作。

另请参见

- “确保按正确顺序应用操作”一节第 100 页
- “重新发送丢失或损坏的消息”一节第 101 页

控制消息大小

以下部分讨论了消息代理 (dbremote) 中的消息编码和压缩方案。

消息代理提供了以下编码和压缩功能：

- **兼容性** 可以将系统设置为与先前版本的 SQL Anywhere 兼容。请参见“升级 SQL Remote”一节 《SQL Anywhere 11 - 更改和升级》。
- **压缩** 可以选择消息的压缩级别。

  消息大小会影响消息通过系统的效率。消息系统处理经过压缩的消息要比处理未经压缩的消息效率更高。但是，压缩可能需要大量的时间。请参见“compression 选项 [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》。
- **编码** SQL Remote 会对消息进行编码，以确保消息经过消息系统时不会损坏。可以自定义编码方案，以提供额外的功能。请参见“编码：防止消息损坏”一节第 103 页。

有关 SQL Remote 对最大消息长度的要求的信息，请参见“消息代理 (dbremote) 的要求”一节 第 87 页。

编码：防止消息损坏

SQL Remote 会对消息进行编码，以确保消息经过消息系统时不会损坏。SQL Remote 的缺省消息编码行为如下：

- 如果消息系统可支持二进制消息格式，则不对消息进行编码。
- 如果消息系统（例如 SMTP）需要基于文本的消息格式，则编码 DLL (*dbencod.dll*) 会在发送消息之前将其转换为文本格式。在接收端，将使用相同的 DLL 对消息格式进行解码。

  可以自定义编码方案以提供额外功能。请参见“创建自定义编码方案”一节第 103 页。
- 如果数据库选项 compression 已设置为 -1，则与版本 5 兼容的编码可用于所有消息系统。请参见“升级 SQL Remote”一节 《SQL Anywhere 11 - 更改和升级》。

创建自定义编码方案

要实现自定义编码方案，可以建立自定义编码 DLL。可以使用此自定义 DLL 应用特定消息系统所需的特殊功能，或收集统计信息，例如向各个用户发送的消息数。

头文件 *dbrmt.h* 安装在 *install-dir* 目录下。此文件包含可用于构建自定义编码方案的应用程序编程接口。

要使用自定义 DLL，请将消息控制参数 encode_dll 设置为自定义 DLL 的完整路径值。例如：

```
SET REMOTE FTP OPTION "Public"."encode_dll" = 'c:\\sany11\\Bin32\\custom.dll';
```

编码和解码必须兼容
如果要使用自定义编码，必须确保接收端也有该 DLL，并且该 DLL 能够正确对消息进行解码。

另请参见

- “SET REMOTE OPTION 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》

SQL Remote 消息系统

SQL Remote 使用一个或多个基础消息系统在各数据库间交换数据。SQL Remote 支持以下消息系统：

- **文件共享** 不要求安装额外软件的简单系统。请参见“FILE 消息系统”一节第 108 页。
- **FTP** Internet 文件传输协议。请参见“FTP 消息系统”一节第 110 页。
- **SMTP/POP** Internet 电子邮件协议。请参见“SMTP 消息系统”一节第 111 页。

为用户指派 REMOTE 或 CONSOLIDATE 权限时选择消息系统。请参见“授予 REMOTE 权限”一节第 26 页和“授予 CONSOLIDATE 权限”一节第 28 页。

用于 SQL Remote 系统的每个消息系统都有一些必须设置的控制参数和其它设置。

并非所有消息系统都可在所有操作系统上获得支持。有关支持的操作系统的完整列表，请参见“支持的平台”一节 《SQL Anywhere 11 - 简介》。

设置消息系统

必须先设置发布者的地址，才能使用消息系统。

每个消息类型定义都包括消息系统类型名（**FILE**、**FTP** 或 **SMTP**）和该消息类型下发布者的地址。

随消息类型定义提供的地址与数据库的发布者 ID 关系紧密。

抽取实用程序 (dbxtract)

抽取实用程序 (dbxtract) 和 [**抽取数据库向导**] 在创建远程数据库时，将统一数据库的发布者地址用作返回地址。它还会被消息代理 (dbremote) 用来为 FILE 系统标识进来的消息的位置。

另请参见

- “FILE 消息系统”一节第 108 页
- “FTP 消息系统”一节第 110 页
- “SMTP 消息系统”一节第 111 页

创建消息类型

♦ **添加消息类型 (Sybase Central)：**

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 在左窗格中，展开 [**SQL Remote 用户**] 目录。
3. 在右窗格中，单击 [**消息类型**] 选项卡。
4. 选择 [**文件**] » [**新建**] » [**消息类型**]。
5. 在 [**您要给新消息类型指定什么名称**] 字段中键入消息类型的名称。该名称应与已安装在 SQL Anywhere 安装目录中的某个消息类型 DLL 相对应。单击 [**下一步**]。

6. 在 [**发布者地址是什么**] 字段中键入发布者地址。单击 [**完成**]。

♦ 创建消息类型 (SQL)：

1. 验证您已创建了该消息类型下的发布者地址。
2. 执行 CREATE REMOTE MESSAGE TYPE 语句。

```
CREATE REMOTE MESSAGE TYPE message-type ADDRESS publisher-address
```

例如：

```
CREATE REMOTE MESSAGE TYPE FILE
ADDRESS 'company';
```

在 Windows Mobile 上创建远程消息类型

如果安装了 Windows Mobile 服务，则可以从 Sybase Central 将 SQL Remote 设置为能够进行 ActiveSync 同步。此选项将存放 FILE 消息链接消息的目录设置为 ActiveSync 目录。将 Windows Mobile 设备与台式计算机连接后，ActiveSync 会使台式计算机上 ActiveSync 目录中的文件与 Windows Mobile 设备上 ActiveSync 目录中的文件保持同步。

♦ 设置 SQL Remote ActiveSync 同步

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 从 [**工具**] 中选择 [**SQL Anywhere 11**] » [**编辑 Windows Mobile 消息类型**]。

另请参见

- “CREATE REMOTE MESSAGE TYPE 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “变更消息类型”一节第 106 页
- “删除消息类型”一节第 107 页

变更消息类型

要更改发布者的地址，请变更其消息类型。无法更改现有消息类型的名称，而必须先将其删除，然后以新名称创建新的消息类型。

♦ 变更远程消息类型 (Sybase Central)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 在左窗格中，展开数据库的 [**SQL Remote 用户**] 目录。
3. 在右窗格中，单击 [**消息类型**] 选项卡。
4. 在右窗格中，右击要变更的消息类型，然后选择 [**属性**]。
5. 更新消息类型属性并单击 [**确定**]。

♦ **变更远程消息类型 (SQL)**

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 确保已确定了该消息类型下的新发布者地址。
3. 执行 ALTER REMOTE MESSAGE TYPE 语句。

另请参见

- “ALTER REMOTE MESSAGE TYPE 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “创建消息类型”一节第 105 页
- “删除消息类型”一节第 107 页

删除消息类型

删除消息类型会将发布者地址从定义中删除。

♦ **删除消息类型 (Sybase Central)**

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 在左窗格中，展开数据库的 [**SQL Remote 用户**] 目录。
3. 在右窗格中，单击 [**消息类型**] 选项卡。
4. 在右窗格中，右击要删除的消息类型，然后选择 [**删除**]。
5. 单击 [**是**]。

♦ **删除消息类型 (SQL)**

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 执行 DROP REMOTE MESSAGE TYPE 语句。

另请参见

- “DROP REMOTE MESSAGE TYPE 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “创建消息类型”一节第 105 页
- “变更消息类型”一节第 106 页

设置远程消息类型控制参数

消息控制参数保存在数据库中。利用以下步骤设置控制参数。

♦ **设置消息控制参数 (SQL)**

- 执行 SET REMOTE OPTION 语句。

  例如，对于用户 myuser 的 FTP 链接，以下语句将 FTP 主机设置为 *ftp.mycompany.com*：

  ```
  SET REMOTE FTP OPTION myuser.host = 'ftp.mycompany.com';
  ```

  请参见“SET REMOTE OPTION 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。

♦ **查看消息链接参数 (SQL)**

- 查询 SYSREMOTEOPTION 系统视图。

  例如：

  ```
  SELECT * from SYSREMOTEOPTION;
  ```

  请参见“SYSREMOTEOPTION 系统视图”一节 《SQL Anywhere 服务器 - SQL 参考》。

存储在磁盘上的消息链接参数

SQL Remote 的早期版本将消息链接参数存储在数据库以外。建议不要对消息链接参数进行外部存储。

消息链接控制参数存储在以下位置：

- **Windows** 存储在注册表的以下位置中：

  ```
  \\HKEY_CURRENT_USER
     \Software
        \Sybase
           \SQL Remote
  ```

  每个消息链接的参数都存储在 SQL Remote 键下的一个键中，键名为该消息链接的名称（例如，4、*SMTP*）。

- **Unix** FILE 系统目录设置存储在 SQLREMOTE 环境变量中。

SQL Remote 环境变量存储的路径可以替代 FILE 消息传递系统的控制参数之一。

当消息代理 (dbremote) 装载一个消息链接时，该链接将使用当前发布者的设置；如果未指定设置，则该链接将使用发布者所属组的设置。

在 Windows 上，第一次运行支持将消息链接参数存储在数据库中的消息代理 (dbremote) 时，该消息代理会将链接选项从注册表复制到数据库中。

另请参见

- “SET REMOTE OPTION 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》

FILE 消息系统

通过使用 FILE 消息系统，即使未安装电子邮件或 FTP 系统，也可以使用 SQL Remote。

FILE 消息系统中的地址

FILE 消息系统是一个简单的 FILE 共享系统。远程用户的 FILE 地址是用于向其中写入用户所有消息的子目录。要检索消息，应用程序将从包含用户文件的目录中读取这些消息。返回消息将发送到统一数据库的地址（写入到统一数据库的目录中）。

作为 Windows 服务运行时，运行消息代理 (dbremote) 所使用的帐户必须具有读写所有必要目录的权限。如果未指派正确的权限，则消息代理无法访问网络驱动器。

地址的根目录

通常，FILE 消息系统地址是所有 SQL Remote 用户都可以访问（无论是通过调制解调器还是局域网）的共享目录的子目录。每个用户都应有一个指向该共享目录的注册表条目、初始化文件条目或 SQLREMOTE 环境变量。

也可以使用 FILE 系统将消息放置在统一计算机和远程计算机上的目录中。可以使用简单的文件传输机制来交换文件并完成复制。

FILE 消息控制参数

FILE 消息系统使用以下控制参数：

- **Directory** 存储消息的目录。该设置是 SQLREMOTE 环境变量的替换值。
- **Debug** 此参数的设置为 YES 或 NO。缺省值为 NO。当设置为 YES 时，将显示由 FILE 链接执行的所有 FILE 系统调用。
- **Encode_dll** 如果正在使用自定义编码方案，则必须将此参数设置为所创建的自定义编码 DLL 的完整路径。请参见“控制消息大小”一节第 103 页。
- **invalid_extensions** 在消息传递系统中生成文件时，不想让消息代理 (dbremote) 使用的、文件扩展名的以逗号分隔的列表。
- **Unlink_delay** 在上次尝试删除文件失败的情况下，再次尝试删除前等待的秒数。如果未定义 unlink_delay 的值，则缺省行为将被设置为在第一次尝试失败后暂停 1 秒，在第二次尝试失败后暂停 2 秒，在第三次尝试失败后暂停 3 秒，在第四次尝试失败后暂停 4 秒。

Windows Mobile 和 ActiveSync

消息代理 (dbremote) 在 *C:\My Documents\Synchronized Files* 中搜索 FILE 链接。在运行统一数据库的计算机上，用于 FILE 链接的 SQLREMOTE 环境变量或目录消息链接参数应设置为以下目录，其中 *userid* 和 *Windows-mobile-device-name* 被设置为适当的值：

```
%SystemRoot%\Profiles\userid\Personal\Windows-mobile-device-name
\Synchronized Files
```

通过此系统，ActiveSync 自动在统一数据库计算机和 Windows Mobile 设备之间同步消息文件。

要验证 FILE 同步是否被激活，请选中 [**移动设备**] » [**工具**] » [**ActiveSync 选项**]。

有关设置消息链接参数的信息，请参见“FILE 消息系统”一节第 108 页。

另请参见

- “FTP 消息系统”一节第 110 页

FTP 消息系统

在 FTP 消息系统中，消息存储在 FTP 主机上根目录下的目录中。FTP 主机和根目录由保存在注册表或初始化文件中的消息系统控制参数来指定，而各个用户的地址就是保存其消息的子目录。

有关支持 FTP 的操作系统的列表，请参见“支持的平台”一节 《SQL Anywhere 11 - 简介》。

FTP 消息控制参数

FTP 消息系统使用以下控制参数：

- **host** 运行 FTP 服务器的计算机的主机名。此参数可以是主机名（如 FTP.ianywhere.com）或 IP 地址（如 192.138.151.66）。
- **user** 用于访问 FTP 主机的用户名。
- **password** 用于访问 FTP 主机的口令。
- **root_directory** 存储消息的 FTP 主机站点内的根目录。
- **port** 用于 FTP 连接的 IP 端口号。此参数通常不是必需项。
- **debug** 可将此参数设置为 YES 或 NO。缺省值为 NO。当设置为 YES 时，将显示调试输出。
- **active_mode** 可将此参数设置为 YES 或 NO。缺省设置为 NO（被动模式）。
- **reconnect_retries** 失败前链接尝试打开与服务器的套接字的次数。缺省值为 4。当您设置此参数时，只会影响重新连接。FTP 链接建立的初始连接不受影响。
- **reconnect_pause** 每次连接尝试间停顿的时间（以秒为单位）。缺省设置为 30 秒。当您设置此参数时，只会影响重新连接。FTP 链接建立的初始连接不受影响。
- **suppress_dialogs** 可将此参数设置为 TRUE 或 FALSE。如果设置为 TRUE，则在尝试连接到 FTP 服务器失败后，将不会出现 [**连接**] 窗口，相反会生成错误。
- **invalid_extensions** 在消息传递系统中生成文件时不想让 dbremote 使用的文件扩展名的逗号分隔列表。
- **encode_dll** 如果实现了自定义编码方案，则必须将此参数设置为所创建的自定义编码 DLL 的完整路径。

另请参见

- “控制消息大小”一节第 103 页

解决 FTP 问题

FTP 消息链接的大部分问题都是由网络系统问题造成的。本节提供了一系列可用于解决问题的测试。

- **设置 DEBUG 消息控制参数** 查看调试输出以确定是否正在连接到 FTP 服务器。如果正在进行连接，则调试输出应指出哪些 FTP 命令执行失败。

- **对 FTP 服务器执行 Ping 命令** 如果 FTP 链接无法连接到 FTP 服务器，则测试您的系统网络配置。例如，可运行以下命令：

  ```
  ping FTP-server-name
  ```

  应返回 FTP 服务器的 IP 地址和强制 FTP 服务器回应的时间（往返时间）。如果无法强制 FTP 服务器回应，则表示网络配置有问题，您应与网络管理员联系。

- **检查被动模式是否能用** 如果 FTP 链接已经连接到该 FTP 服务器，但无法打开数据连接，则请检查是否可以使用被动模式在 FTP 客户端与该服务器之间传送数据。

  被动模式是首选传送模式，也是 FTP 消息链接的缺省模式。在被动模式下，所有数据传送连接都由客户端（此处为消息链接）启动。在主动模式下，所有数据连接都由 FTP 服务器启动。如果您的 FTP 服务器位于配置不正确的防火墙后面，则您可能无法使用缺省的被动传送模式，因为防火墙会阻止通过 FTP 控制端口以外的端口与 FTP 服务器建立套接字连接。

  通过使用允许在**主动**和**被动**之间选择传送模式的 FTP 用户程序，可将传送模式设置为被动，然后尝试上载或下载文件。如果您所使用的客户端在不使用主动模式的情况下无法传送文件，则应将防火墙和 FTP 服务器重新配置成可以使用被动模式进行传送，或将 active_mode 消息控制参数设置为 YES。主动模式传送可能并不适用于所有网络配置。例如，如果您的客户端位于 IP 伪装网关后面，则根据所用网关软件，进来的连接可能会失败。

- **检查权限和目录结构** 如果已经与 FTP 服务器建立了连接，但在获取目录列表或处理文件时遇到问题，则检查权限设置是否正确，所需目录是否存在。

  使用 FTP 程序登录到 FTP 服务器。转到 root_directory 参数中存储的目录。如果所需目录没有出现，则表示 root_directory 控制参数可能设置错误，或所需目录可能不存在。

  通过获取消息目录中的文件和将文件上载到统一数据库目录来对权限进行测试。如果返回错误，则表示 FTP 服务器权限设置错误。

SMTP 消息系统

使用 SMTP 系统时，SQL Remote 通过 Internet 邮件发送消息。将消息编码成文本格式，然后通过电子邮件消息将其发送到目标数据库。使用 SMTP 服务器发送消息并从 POP 服务器中检索它们。

有关支持 SMTP 的操作系统的列表，请参见“支持的平台”一节 《SQL Anywhere 11 - 简介》。

SMTP 地址和用户 ID

要使用 SQL Remote 和 SMTP 消息系统，则参与到系统中的每个数据库都需要 SMTP 地址以及 POP3 用户 ID 和口令。它们是不同的标识符：SMTP 地址是每个消息的终点，而 POP3 用户 ID 和口令是用户在连接到其电子邮件服务器时输入的名称和口令。

建议使用单独的电子邮件帐户
建议使用单独的 POP 电子邮件帐户来发送和接收 SQL Remote 消息。请参见“共享 SMTP/POP 地址”一节第 112 页。

SMTP 消息控制参数

在消息代理 (dbremote) 连接到消息系统以发送或接收消息之前，用户必须已在其计算机上设置了一组控制参数，否则该用户将被提示指定所需信息。仅在第一次连接时需要这些信息。这些信息将被保存起来，并在以后进行连接时用作缺省条目。

SMTP 消息系统使用以下控制参数：

- **local_host** 本地计算机的名称。它在 SQL Remote 无法确定本地主机名的计算机上会有用。启动与任何 SMTP 服务器之间的会话时，都需要本地主机名。在大多数网络环境中，本地主机名可以自动确定，因而不需要此条目。
- **TOP_supported** SQL Remote 在枚举进来的消息时，使用名为 TOP 的 POP3 命令。并非所有 POP 服务器都支持 TOP 命令。当将 TOP_supported 参数设置为 NO 时，SQL Remote 使用 RETR 命令，该命令效率较低，但可与所有 POP 服务器协同工作。缺省值为 YES。
- **smtp_authenticate** 决定 SMTP 链接是否对用户进行验证。缺省值为 YES。将此参数设置为 NO 以关闭 SMTP 验证。
- **smtp_userid** 用于 SMTP 验证的用户 ID。缺省情况下，此参数与 pop3_userid 参数使用相同的值。只有在用户 ID 与 POP 服务器上的用户 ID 不同时，才需要设置 smtp_userid 参数。
- **smtp_password** 用于 SMTP 验证的口令。缺省情况下，此参数与 pop3_password 参数使用相同的值。只有在用户 ID 与 POP 服务器上的用户 ID 不同时，才需要设置 smtp_password。
- **smtp_host** 运行 SMTP 服务器的计算机的名称。它对应于 SMTP/POP3 登录窗口中的 SMTP 主机字段。
- **pop3_host** 运行 POP 主机的计算机的名称。通常与 SMTP 主机的名称相同。它对应于 SMTP/POP3 登录窗口中的 POP3 主机字段。
- **pop3_userid** 用于检索邮件的用户 ID。该参数指 POP 用户 ID，它对应于 SMTP/POP3 登录窗口中的 [用户 ID] 字段。您必须向 POP 主机管理员索取用户 ID。
- **pop3_password** 用于检索邮件的口令。它对应于 SMTP/POP3 登录窗口中的口令字段。
- **Debug** 当设置为 YES 时，将显示所有 SMTP 和 POP3 命令及其响应。此信息可用于解决 SMTP/POP 支持问题。缺省值是 NO。
- **Suppress_dialogs** 如果将此参数设置为 true，则在尝试连接到邮件服务器失败后，将不会出现 [**连接**] 窗口，相反会生成错误。
- **encode_dll** 如果实现了自定义编码方案，必须将此参数设置为所创建自定义编码 DLL 的完整路径。

请参见“控制消息大小”一节第 103 页。

共享 SMTP/POP 地址

应为 SQL Remote 消息使用单独的电子邮件帐户。不建议您通过用于个人或企业电子邮件消息的相同电子邮件帐户来发送和接收 SQL Remote 消息。

如果需要为 SQL Remote 消息和常规电子邮件消息共享相同的电子邮件帐户，则必须确保电子邮件程序不会从邮件服务器下载和删除所有消息（包括 SQL Remote 电子邮件和个人消息）。必须配置电子邮件程序，以使其在下载常规电子邮件消息时不会变更或删除 SQL Remote 消息。SQL Remote 消息包含主题 ---SQL Remote--。

解决 SMTP 链接问题

如果无法使 SMTP 链接正常工作，则使用同一帐户和口令从运行消息代理 (dbremote) 的同一台计算机连接到 SMTP/POP3 服务器。使用支持 SMTP/POP3 的 Internet 电子邮件程序。一旦 SMTP 消息链接正在工作，请禁用此程序。

检查电子邮件是否在正常工作

如果 SQL Remote 消息的收发不正常，并且您正在使用电子邮件消息系统，则应确认电子邮件在两台计算机间的工作是否正常。

备份 SQL Remote 系统

SQL Remote 复制过程取决于对事务日志中的操作的访问，以及对旧事务日志的访问。您实施的所有备份策略必须包括 SQL Remote 的事务日志的维护。

消息代理 (dbremote) 必须具有对当前事务日志和旧事务日志的访问权限，直到其不再被需要。

当所有远程数据库已收到事务日志所包含的消息，并确认这些消息已成功应用时，统一数据库将不再需要其事务日志。

当统一数据库已收到事务日志所包含的消息，并确认这些消息已成功应用时，远程数据库将不再需要其事务日志。

备份远程数据库

针对远程数据库，您需要确定是否：

- **将复制统一数据库作为备份方法** 备份过程在远程数据库上并不像在统一数据库上那样重要。可以将复制统一数据库作为数据备份方法。

  如果选择此方法，则应创建维护远程数据库事务日志的策略。请参见“维护远程数据库的事务日志”一节第 114 页。

- **为远程数据库创建备份策略** 如果在远程数据库上所做的更改至关重要，则需要为远程数据库创建包括事务日志维护在内的备份策略。请参见“备份远程数据库”一节第 115 页。

备份统一数据库

必须拥有统一数据库的包括事务日志维护在内的备份策略。请参见“防止统一数据库发生介质故障”一节第 116 页。

备份实用程序 (dbbackup) 和消息代理 (dbremote) 的 -x 选项

在数据库中，您决不应该同时运行带 -x 选项的消息代理 (dbremote) 和备份实用程序 (dbbackup)。

-x 选项可用来管理用于复制的事务日志。-x 选项可以确保消息代理能够访问旧的事务日志，并在不再需要这些事务日志时将其删除。-x 选项不备份事务日志。

备份实用程序 (dbbackup) 可用于备份当前事务日志。当使用 -r 和 -n 选项运行备份实用程序 (dbbackup) 时，备份实用程序会将当前事务日志备份到备份目录，然后重命名并重新启动该当前事务日志。备份实用程序 (dbbackup) 假定当前事务日志就是上次备份之后重命名并重新启动的那个事务日志。

如果尝试在同一数据库中同时运行消息代理的 -x 选项和备份实用程序 (dbbackup)，则它们会相互干扰。当二者同时运行时，事务日志可能会丢失。

在未处于备份中的远程数据库上，只运行带 -x 选项的消息代理 (dbremote)。

维护远程数据库的事务日志

当通过复制统一数据库来备份远程数据库时，使用以下过程来维护远程数据库的事务日志。也就是说，您*并非*在远程数据库及其事务日志上运行备份实用程序 (dbbackup)。

小心
*不要*在正处于备份中的数据库上运行带 -x 选项的消息代理 (dbremote)。

♦ **维护远程数据库的事务日志**

1. 在远程数据库上，运行带 -x 选项的消息代理 (dbremote)，并指定事务日志的大小。当事务日志超过指定的大小时，此选项会使消息代理 (dbremote) 重命名事务日志并将其重新启动。

   以下命令在事务日志大于 1 MB 时将其删除：

   ```
   dbremote -x 1M  -c "UID=ManagerSteve;PWD=sql;DBF=c:\mydata.db"
   ```

2. 在远程数据库上，将 delete_old_logs 选项设置为 On。当不再需要用于复制的旧事务日志文件时，设置 delete_old_logs 选项会使这些文件自动被消息代理 (dbremote) 删除。

   当所有预订者都确认其已收到并成功地应用了事务日志文件中所记录的全部更改时，便不再需要该事务日志了。可以为 PUBLIC 组或者仅为消息代理 (dbremote) 连接字符串中包含的用户设置 delete_old_logs 选项。

   以下语句将公共 delete_old_logs 选项设置为删除十天以前创建的日志：

   ```
   SET OPTION PUBLIC.delete_old_logs = '10 days';
   ```

备份远程数据库

利用以下过程备份远程数据库。此过程包括 SQL Remote 使用事务日志的维护策略。*不要*使用此过程运行带 -x 选项的消息代理 (dbremote)。

♦ **使用备份实用程序 (dbbackup) 备份远程数据库**

1. 完全备份远程数据库。
   a. 以具有 DBA 权限的用户身份连接到数据库。
   b. 使用 -r 和 -n 选项运行 dbbackup。

      例如，假定备份目录为 *e:\archive*，数据库文件位于 *c:\live* 目录中，并且其相应事务日志文件位于 *d:\live* 目录中：

      ```
      dbbackup -r -n -c "UID=DBA;PWD=sql;DBF=c:\live\remotedatabase.db" e:
      \archive
      ```

      d:\live 目录中的事务日志未被完全备份变更。
   c. 将位于 *e:\archive* 目录中的备份文件复制到站外驱动器或 DVD。
   d. 使用以下命令运行具有当前事务日志文件访问权限的消息代理 (dbremote)：

      ```
      dbremote -c "UID=DBA;PWD=sql;DBF=c:\live\database.db" d:\live
      ```

   小心
   *不要*在正处于备份中的数据库上运行带 -x 选项的消息代理 (dbremote)。

2. 设置备份实用程序 (dbbackup) 以进行远程数据库事务日志的增量备份。

a. 以具有 DBA 权限的用户身份连接到数据库。

b. 使用 -r、-n 和 -t 选项运行 dbbackup。

例如：

```
dbbackup -r -n -t -c "UID=DBA;PWD=sql;DBF=c:\live\remotedatabase.db" e:
\archive
```

c. 使用以下命令运行具有当前事务日志文件访问权限的消息代理 (dbremote)：

```
dbremote -c "UID=DBA;PWD=sql;DBF=c:\live\remotedatabase.db" d:\live
```

防止统一数据库发生介质故障

保护 SQL Remote 复制系统使其不会发生介质故障：

- **仅复制已备份的事务** 发送只包含已备份事务的消息。通过只发送已备份的事务可保护复制系统，使其不会发生事务日志介质故障。可通过以下操作实现此目的：
  - 使用 -u 选项运行消息代理 (dbremote)。使用 -u 选项运行消息代理 (dbremote) 时，只将已备份且已提交的事务封装在消息中进行发送。

    如果将数据备份到其它站点，则 -u 选项还可以防止整个站点出现故障。请参见“消息代理 (dbremote)”一节第 140 页。
- **使用事务日志镜像** 使用事务日志镜像可以防止因事务日志设备出现介质故障而造成的损失。请参见“事务日志镜像”一节 《SQL Anywhere 服务器 - 数据库管理》。
- **不要在统一数据库上运行带 -x 选项的消息代理 (dbremote)** 决不要针对正处于备份中的数据库运行带 -x 选项的消息代理 (dbremote)。-x 选项将维护用于复制而非备份或恢复的事务日志。要备份统一数据库，请参见“备份统一数据库”一节第 116 页。

备份统一数据库

通过完全备份统一数据库和事务日志来备份您的统一数据库，然后对事务日志进行增量备份。

♦ 备份 SQL Remote 统一数据库

1. 对统一数据库及其事务日志进行完全备份。
   a. 以具有 DBA 权限的用户身份连接到数据库。
   b. 使用 -r 和 -n 选项运行 dbbackup。

   例如：

   ```
   dbbackup -r -n -c "UID=DBA;PWD=sql;DBF=c:\live\database.db" e:\archive
   ```
2. 对统一数据库的事务日志进行增量备份。备份事务日志时，选择重命名并重新启动事务日志。
   a. 以具有 DBA 权限的用户身份连接到数据库。
   b. 使用 -r、-n 和 -t 选项运行 dbbackup。

   例如：


```
dbbackup -r -n -t -c "UID=DBA;PWD=sql;DBF=c:\live\database.db" e:
\archive
```

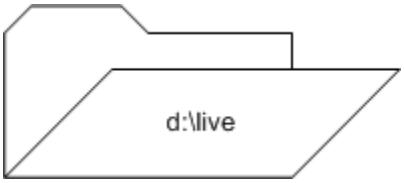
3. 运行具有当前事务日志访问权限的消息代理 (dbremote)。

   例如：

```
dbremote -c "UID=DBA;PWD=sql;DBF=c:\live\database.db" d:\live
```

小心
不要 在正处于备份中的数据库上运行带 -x 选项的消息代理 (dbremote)。

下图说明了 *c:\live* 目录中的一个名为 *database.db* 的数据库，该数据库具有一个名为 *database.log* 的事务日志，位于 *d:\live* 目录中。



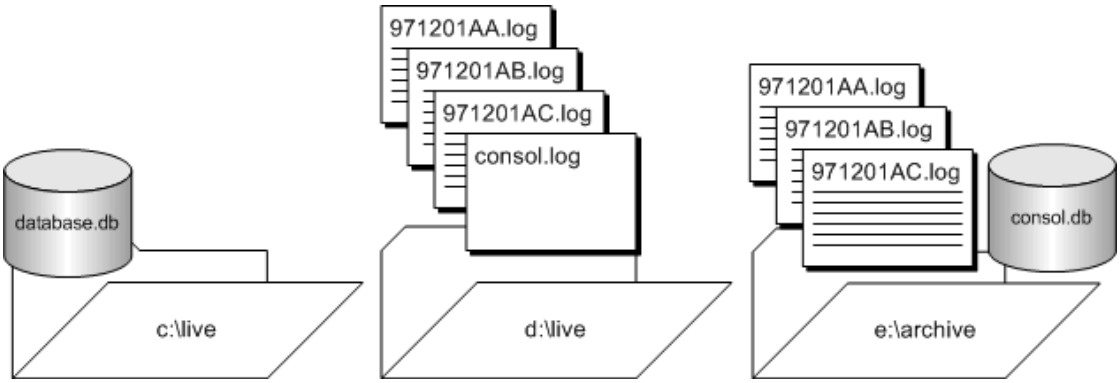
当使用 -r 和 -n 选项将事务日志备份到备份目录 *e:\archive* 中，以重命名并重新启动事务日志时，备份实用程序 (dbbackup) 将执行以下任务：

1. 将当前事务日志文件重命名为 *971201xx.log*，其中 *xx* 是从 *AA* 到 *ZZ* 的顺序字符。
2. 通过创建一个名为 *971201xx.log* 的备份文件，将事务日志文件备份到备份目录中。

   8.0.1 版之前的旧事务日志名称
   在 SQL Anywhere 的 8.0.1 版之前，旧事务日志文件被命名为 *yymmdd01.log*、*yymmdd02.log* 等。为了存储更多的旧事务日志，对命名方式进行了更改。由于消息代理 (dbremote) 扫描指定目录中的所有文件时不考虑它们的名称，因此该名称更改不会影响现有应用程序。

3. 启动一个新事务日志，如 *database.log*。

几次备份之后，活动目录和档案目录都将包含一组有序的事务日志。



另请参见

- “防止统一数据库发生介质故障”一节第 116 页
- “在备份过程中重命名事务日志的备份副本”一节 《SQL Anywhere 服务器 - 数据库管理》
- “备份实用程序 (dbbackup)”一节 《SQL Anywhere 服务器 - 数据库管理》

手工恢复统一数据库

以下过程介绍了如何通过将各个事务日志应用到数据库来恢复统一数据库。要使 SQL Anywhere 数据库服务器自动恢复统一数据库，请参见“自动恢复统一数据库”一节第 121 页。

♦ 使用 -a 选项恢复数据库

1. 创建数据库和事务日志文件的副本。此过程假定先前已备份了数据库文件，并且现在可用（例如在磁带上）。
2. 创建一个临时目录。
3. 将数据库 (*.db*) 文件的最新备份（而*非*事务日志文件）从磁带恢复到临时目录中。

   在该临时目录中：

   a. 启动数据库的备份副本。

   b. 使用 -a 选项应用旧事务日志。

   c. 关闭数据库。

   d. 使用当前事务日志和 -a 选项启动数据库，以应用事务并更新数据库文件。

   e. 关闭数据库。

   f. 备份数据库。
4. 将数据库复制到生产目录中。
5. 启动数据库。

   所有新活动将附加到当前事务日志中。

示例：逐一应用事务日志

假设您拥有一个名为 *c:\dbdir\cons.db* 的统一数据库文件、一个事务日志文件 *c:\dbdir\cons.log* 以及一个事务日志镜像文件 *d:\mirdir\cons.mlg*。

假定您每周执行完全备份，并且使用以下命令每天执行增量备份：

```
dbbackup -c "UID=DBA;PWD=sql" -r -n -t e:\backdir
```

此命令将事务日志 *cons.log* 备份到目录 *e:\backdir* 中。这时，该事务日志文件被重命名为 *datexx.log*（其中 *date* 为当前日期，*xx* 为序列中的下一组字母），这样，一个新的事务日志便启动了。目录 *e:\backdir* 随后将使用第三方实用程序进行备份。

在这种情况下，可使用指向已重命名的事务日志文件的可选目录来运行消息代理 (dbremote)。例如：

```
dbremote -c "UID=DBA;PWD=sql" c:\dbdir
```

在每周备份后的第三天，数据库文件因磁盘块错误而损坏。

利用以下过程从介质故障中恢复。

♦ **从 C 驱动器上的介质故障中恢复：**

1. 备份事务日志镜像文件 *d:\mirdir\cons.mlg*。
2. 创建一个用于执行恢复的临时目录。在本示例中，该目录名为 *c:\recover*。
3. 将数据库文件 *cons.db* 的最新备份恢复到 *c:\recover\cons.db* 中。
4. 按顺序应用已重命名的事务日志，如下所示：

```
dbeng11 -a c:\dbdir\dateAA.log c:\recover\cons.db
dbeng11 -a c:\dbdir\dateAB.log c:\recover\cons.db
```

5. 将当前事务日志 *d:\mirdir\cons.log* 复制到恢复目录中，得到 *c:\recover\cons.log*。
6. 使用以下命令启动数据库：

```
dbeng11 c:\recover\cons.db
```

7. 关闭数据库服务器。
8. 从 *c:\recover* 中备份已恢复的数据库和事务日志。
9. 将文件从 *c:\recover* 复制到相应的生产目录：
   - 将 *c:\recover\cons.db* 复制到 *c:\dbdir\cons.db*。
   - 将 *c:\recover\cons.log* 复制到 *c:\dbdir\cons.log* 和 *d:\mirdir\cons.mlg*。
10. 以正常方式重新启动系统。

另请参见

- “-a 数据库选项”一节 《SQL Anywhere 服务器 - 数据库管理》

自动恢复统一数据库

以下过程介绍了如何自动恢复统一数据库。要手工应用事务日志，请参见“手工恢复统一数据库”一节第 119 页。

♦ 使用 -ad 选项恢复数据库

1. 创建数据库和事务日志文件的副本。此过程假定先前已备份了数据库文件，并且现在可用（例如在磁带上）。
2. 将数据库 (*.db*) 文件的最新备份副本（而*非*事务日志文件）从磁带恢复到临时目录中。
3. 在该临时目录中：
   a. 启动数据库，同时使用 -ad 选项应用事务日志。

      当您指定 -ad 选项时，数据库服务器会在指定目录中为数据库查找事务日志。然后，它根据事务日志偏移确定应用日志的正确顺序。
   b. 将当前事务日志复制到临时目录中。
   c. 启动数据库并应用当前事务日志。
   d. 关闭数据库服务器。
   e. 备份数据库和事务日志。
4. 将数据库和事务日志文件复制到适当的生产目录中。
5. 以正常方式重新启动系统。

   所有新活动将附加到当前事务日志中。

示例

假设您拥有一个名为 *c:\dbdir\cons.db* 的统一数据库文件、一个事务日志文件 *c:\dbdir\cons.log* 以及一个事务日志镜像文件 *d:\mirdir\cons.mlg*。

假定您使用以下命令每周执行完全备份：

```
dbbackup -c "UID=DBA;PWD=sql" -r -n e:\backdir
```

假定您还使用以下命令每天执行增量备份：

```
dbbackup -c "UID=DBA;PWD=sql" -r -n -t e:\backdir
```

此命令将事务日志 *cons.log* 备份到目录 *e:\backdir* 中。这时，该事务日志文件被重命名为 *datexx.log*（其中 *date* 为当前日期，*xx* 为序列中的下一组字母），这样，一个新的事务日志便启动了。目录 *e:\backdir* 随后将使用第三方实用程序进行备份。

在这种情况下，可使用指向已重命名的事务日志文件的可选目录来运行消息代理 (dbremote)。例如：

```
dbremote -c "UID=DBA;PWD=sql" c:\dbdir
```

在每周备份后的第三天，数据库文件因磁盘块错误而损坏。

♦ 从 c 驱动器上的介质故障中恢复

1. 替换 *c:* 驱动器。
2. 备份事务日志镜像文件 *d:\mirdir\cons.mlg*。
3. 创建一个用于执行恢复的临时目录。在本示例中，该目录名为 *c:\recover*。
4. 将数据库文件 *cons.db* 的最新备份恢复到 *c:\recover\cons.db* 中。
5. 将备份的事务日志复制到 *c:\dbdir* 中。
6. 应用已重命名的事务日志：

   ```
   dbeng11 c:\recover\cons.db -ad c:\dbdir
   ```

7. 将当前事务日志 *d:\mirdir\cons.log* 复制到恢复目录中，得到 *c:\recover\cons.log*。
8. 使用以下命令启动数据库：

   ```
   dbeng11 c:\recover\cons.db
   ```

9. 关闭数据库服务器。
10. 从 *c:\recover* 中备份已恢复的数据库和事务日志。
11. 将文件从 *c:\recover* 复制到相应的生产目录：
    - 将 *c:\recover\cons.db* 复制到 *c:\dbdir\cons.db*。
    - 将 *c:\recover\cons.log* 复制到 *c:\dbdir\cons.log* 和 *d:\mirdir\cons.mlg*。
12. 以正常方式重新启动系统。

另请参见

- “-ad 数据库选项”一节 《SQL Anywhere 服务器 - 数据库管理》

报告并处理复制错误

SQL Remote 系统上会发生以下错误：

- **未找到行错误** 请参见“未找到行错误”一节第 50 页。
- **参照完整性错误** 请参见“参照完整性错误”一节第 51 页。
- **重复主键错误** 请参见“重复主键错误”一节第 53 页。

缺省情况下，当发生错误时，消息代理 (dbremote) 会在其日志输出窗口中输出错误。消息代理 (dbremote) 在输出消息文件中输出的信息多于在消息窗口中输出的信息。

消息代理 (dbremote) 的消息日志文件包括以下信息：

- 已应用的消息。
- 失败的 SQL 语句。
- 其它错误。

要将错误输出到输出日志文件，请使用 -o 选项运行消息代理 (dbremote)。请参见“消息代理 (dbremote)”一节第 140 页。

发生错误时，可以配置 SQL Remote 以便：

- **运行错误处理过程** 缺省情况下，不调用任何存储过程。但是，可以使用 replication_error 数据库选项来指定在出错时由消息代理 (dbremote) 调用的存储过程。请参见“运行错误处理过程”一节第 123 页。

  例如，可以配置 SQL Remote 以便：

  - 将远程数据库的部分输出日志发送到统一数据库并写入文件。请参见“从远程数据库收集错误”一节第 124 页。
  - 当远程数据库上发生错误时，发送电子邮件通知。请参见“接收关于远程数据库错误的电子邮件通知”一节第 125 页。
- **忽略错误** 可能会存在您不想让消息代理 (dbremote) 报告错误的实例。例如，如果您知道错误会在哪些情况下发生，并确信该错误不会导致数据不一致，则可以选择忽略该错误。请参见“忽略复制错误”一节第 127 页。

运行错误处理过程

设置 replication_error 选项以在发生 SQL 错误时调用某一过程。缺省情况下，当发生 SQL 错误时，不会调用任何过程。

调用的过程必须具有类型为 CHAR、VARCHAR 或 LONG VARCHAR 的一个参数。SQL 错误消息会调用一次该过程，并且导致该错误的 SQL 语句也会调用一次该过程。请参见“replication_error 选项 [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》。

♦ **设置 replication_error 选项**

● 执行以下语句。*remote-user* 是消息代理 (dbremote) 命令中的发布者名称，*procedure-name* 是检测到 SQL 错误时调用的过程。

```
SET OPTION
remote-user.replication_error
= 'procedure-name';
```

从远程数据库收集错误

利用以下过程将远程数据库的部分输出日志发送到统一数据库中。将这些信息写入文件，该文件可以包含系统中部分或全部远程数据库的输出记录信息。

♦ **配置 SQL Remote 以从远程数据库收集输出日志信息**

1. 配置远程数据库以将输出日志信息发送到统一数据库。
   a. 使用带 output_log_send_on_error 选项的 SET REMOTE 语句，以在发生错误时发送日志信息。

      在远程数据库上执行以下命令：

      ```
      SET REMOTE link-name OPTION
      PUBLIC.output_log_send_on_error = 'Yes';
      ```

      消息代理 (dbremote) 在读取到任何以错误指示符 **E** 开头的消息时，都会将输出日志信息发送到统一数据库。请参见“SET REMOTE OPTION 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。
   b. 这是可选步骤。设置带 output_log_send_limit 选项的 SET REMOTE 语句，以限制向统一数据库发送的信息量。output_log_send_limit 选项在输出日志的结束处（即最新条目）指定发送到统一数据库的字节数。缺省值为 5K。

      如果您提供的 output_log_send_limit 值超过了最大消息大小，则 SQL Remote 会覆盖 output_log_send_limit 值，并且只发送满足最大消息大小条件的值。

      在远程数据库上执行以下命令：

      ```
      SET REMOTE link-name OPTION
      PUBLIC.output_log_send_limit = '7K';
      ```

      请参见“SET REMOTE OPTION 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。
2. 配置接收日志信息的统一数据库。

   在统一数据库上，使用 -ro 或 -rt 选项运行消息代理 (dbremote)。

   请参见“消息代理 (dbremote)”一节第 140 页。
3. 这是可选步骤。要测试您的配置，请设置 output_log_send_now 选项以将输出日志信息发送到统一数据库。

   在远程数据库上，将 output_log_send_now 选项设置为 YES。

下次轮询时，远程数据库将发送输出日志信息，然后将 output_log_send_now 选项重置为 NO。

另请参见

- “接收关于远程数据库错误的电子邮件通知”一节第 125 页

接收关于远程数据库错误的电子邮件通知

当远程数据库上发生错误时，利用以下过程发送电子邮件通知。可以使用电子邮件或寻呼系统来接收通知。

♦ 设置 SQL Remote 以发送错误 (SQL) 的电子邮件通知

1. 以用户 Cons 的身份连接到统一数据库。
2. 创建一个可以通过电子邮件通知 DBA 用户有错误发生的存储过程。

   例如，执行以下语句来创建 sp_LogReplicationError 过程：

```
CREATE PROCEDURE cons.sp_LogReplicationError
 ( IN error_text LONG VARCHAR )
BEGIN
 DECLARE current_remote_user CHAR( 255 );
 SET current_remote_user = CURRENT REMOTE USER;
 // Log the error
 INSERT INTO cons.replication_audit
  ( remoteuser, errormsg )
 VALUES
  ( current_remote_user, error_text );
 COMMIT WORK;
 //Now notify the DBA by email that an error has occurred
 // on the consolidated database. The email should contain the error
 // strings that the Message Agent is passing to the procedure.
 IF CURRENT PUBLISHER = 'cons' THEN
  CALL sp_notify_DBA( error_text );
 END IF
END;
```

3. 创建一个管理电子邮件发送的存储过程。

   例如，执行以下语句来创建 sp_notifiy_DBA 过程：

```
CREATE PROCEDURE sp_notify_DBA( in msg long varchar)
BEGIN
 DECLARE rc INTEGER;
 rc=call xp_startmail( mail_user='davidf' );
 //If successful logon to mail
 IF rc=0 THEN
 rc=call xp_sendmail(
   recipient='Doe, John; Smith, Elton',
   subject='SQL Remote Error',
   "message"=msg);
 //If mail sent successfully, stop
  IF rc=0 THEN
   call xp_stopmail()
  END IF
 END IF
END;
```

4. 设置 replication_error 数据库选项，以调用通过电子邮件通知 DBA 有错误发生的过程。

   例如，执行以下语句以在发生错误时调用 sp_LogReplicationError 过程：

```
SET OPTION PUBLIC.replication_error =
 'cons.sp_LogReplicationError';
```

5. 创建一个审计表。

   例如，执行以下语句来创建 replication_audit 表：

```
CREATE TABLE replication_audit (
 id   INTEGER DEFAULT AUTOINCREMENT,
 pub  CHAR(30) DEFAULT CURRENT PUBLISHER,
 remoteuser CHAR(30),
 errormsg LONG VARCHAR,
 timestamp  DATETIME DEFAULT CURRENT TIMESTAMP,
 PRIMARY KEY (id,pub)
);
```

   下表对 replication_audit 表的列进行了说明：

列	说明
pub	数据库的当前发布者（标识已插入了发布者的数据库）。
remoteuser	应用消息的远程用户（标识远程用户所属的数据库）。
errormsg	传递给 replication_error 过程的错误消息。

6. 测试您的过程。

   例如，在统一数据库中插入的行与远程数据库中的行使用相同的主键。将统一数据库中的这一行复制到远程数据库时，将发生主键冲突错误，并且：

   - 远程数据库的消息代理 (dbremote) 会向其输出日志输出以下消息：

```
Received message from "cons" (0-0000000000-0)
SQL statement failed: (-193) primary key for table 'reptable' is not
unique
INSERT INTO cons.reptable( id,text,last_contact )
VALUES (2,'dave','1997/apr/21 16:02:38.325')
COMMIT WORK
```

   - 以下 INSERT 语句将被发送到统一数据库：

```
INSERT INTO cons.replication_audit
 ( id,
   pub,
   remoteuser,
   errormsg,
   "timestamp")
VALUES
 ( 1,
  'cons',
  'sales',
  'primary key for table ''reptable'' is not unique (-193)',
  '1997/apr/21 16:03:13.836');
COMMIT WORK;
```


- 带有以下消息的电子邮件将被发送给 John Doe 和 Elton Smith：

```
primary key for table 'reptable' is not unique (-193)
INSERT INTO cons.reptable( id,text,last_contact ) VALUES
(2,'dave','1997/apr/21 16:02:52.605')
```

另请参见

- “从远程数据库收集错误”一节第 124 页

忽略复制错误

- **忽略复制错误 (SQL)**

- 针对导致已知错误的操作创建一个 BEFORE 触发器。该触发器应发出 REMOTE_STATEMENT_FAILED SQLSTATE (5RW09) 或 SQLCODE (-288) 信号。

  例如，如果您希望忽略在表丢失引用的列时发生的 INSERT 语句错误，则可以创建一个 BEFORE INSERT 触发器，该触发器在引用的列不存在时会发出 REMOTE_STATEMENT_FAILED SQLSTATE 信号。INSERT 语句会失败，但消息代理 (dbremote) 输出日志中将不报告这一失败。请参见“远程语句失败”一节 《错误消息》。

安全性

使用以下可用于保护您的数据的功能。

- **REMOTE DBA 特权** 建议以具有 REMOTE DBA 权限的用户身份连接到消息代理 (dbremote)。请参见“授予 REMOTE DBA 权限”一节第 30 页。
- **数据库加密** 可以使用 -ek 选项来加密您的数据库。请参见“抽取实用程序 (dbxtract)”一节第 147 页。
- **消息加密** 消息代理 (dbremote) 使用简单的加密算法，以防止消息被偶然窃取。不过，该加密方案并非专用于提供防止刻意解密行为的全面防护。有关数据库加密的信息，请参见“加密和解密数据库”一节 《SQL Anywhere 服务器 - 数据库管理》。

升级和重新同步

升级 SQL Remote 系统时应小心谨慎。可以使用以下任何一种方式来升级 SQL Remote 系统：

- **升级软件** 有关升级 SQL Remote 的信息，请参见“升级 SQL Remote”一节 《SQL Anywhere 11 - 更改和升级》。
- **更改数据库模式** 要更改数据库模式，可以：
  - **使用直通模式** 直通模式允许将模式更改发送到 SQL Remote 系统中的某些或全部数据库，但需要仔细地规划和执行。请参见“SQL Remote 直通模式”一节第 130 页。
  - **重新同步预订** 重新同步包括将新的数据副本复制到远程数据库。当存在多个远程数据库时，重新同步可能是一个费时的过程，不仅会造成工作中断而且还可能丢失数据。请参见“重新同步预订”一节第 133 页。

在正在运行的系统上应避免进行的更改

不应对已部署且正在运行的 SQL Remote 系统进行以下更改，但已注明的情况除外：

- **更改发布者** 如果在已部署且正在运行的 SQL Remote 系统中的统一数据库上更改发布者的用户名，则会出现问题。如果需要更改统一数据库中发布者的用户名，则必须关闭 SQL Remote 系统并重新同步所有远程用户。

  如果更改远程数据库中发布者的用户名，则会导致所有涉及该远程数据库的预订出现问题（包括信息丢失）。如果需要更改远程数据库中发布者的用户名，则关闭远程数据库并重新同步远程用户。请参见“重新同步预订”一节第 133 页。
- **对表进行限制性更改** 不能对表进行限制性更改。例如，不能删除某列或变更某列以禁止使用 NULL 值，因为消息可能存在于引用这些列的系统中。
- **对表进行允许的更改** 可以使用直通模式来进行允许的更改。使用直通模式更改远程数据库模式和发布。允许的更改包括添加新表或新列、添加新用户、重新同步用户、删除用户以及为远程用户更改地址、消息类型或发送频率。请参见“SQL Remote 直通模式”一节第 130 页。
- **变更发布** 必须同时在统一数据库和远程数据库上维护发布定义。如果在正在运行的 SQL Remote 系统中变更发布，则会导致复制错误以及复制系统中的数据丢失。
- **删除预订** 可以删除预订，但必须使用直通模式删除远程数据库中的数据。请参见“SQL Remote 直通模式”一节第 130 页。
- **卸载和重装数据库** 必须确保事务日志已正确维护。请参见“重建参与同步或复制的数据库”一节 《SQL Anywhere 服务器 - SQL 的用法》。
- **在多层系统中进行更改** 有关在多层系统中重新抽取数据库模式的信息，请参见“为多层系统抽取数据库”一节第 83 页。

  有关使用直通模式进行模式更改的信息，请参见“直通模式限制”一节第 130 页。

SQL Remote 直通模式

使用直通模式将标准 SQL 语句传递到可执行这些语句的远程数据库中。

使用直通模式在正在运行的 SQL Remote 系统中完成以下任务：

- 添加新用户。
- 将用户重新同步。
- 从系统中删除用户。
- 更改远程用户的地址、消息类型或频率。
- 在表中添加列。

小心

- SQL Remote 依赖于系统中包含相同对象的各个数据库；当某个表在一些站点上被更改，但在其它站点上未被更改时，尝试复制数据更改将会失败。在正在运行的 SQL Remote 系统上执行其它模式更改可能会产生问题。请参见“在正在运行的系统上应避免进行的更改”一节第 129 页。
- 应始终在已预订了远程数据库副本的情况下，在统一数据库的副本上测试您的直通操作。决不要在生产数据库上运行未经测试的直通脚本。
- 应始终用所有者名称限定对象名称。PASSTHROUGH 语句不会在远程数据库上由同一个用户名来执行。不含所有者名称限定符的对象名称可能无法正确解析。

直通模式限制

- **直通仅在层次中的一个层次上起作用** 在多层 SQL Remote 系统中，直通语句在当前层的下一层上工作，这一点很重要。在多层系统中，必须在每个统一数据库上为该统一数据库下面的那一层输入直通语句。
- **调用过程** 当在直通模式下使用 CALL 或 EXEC 语句调用存储过程时：
  - 该过程必须存在于调用直通命令的统一数据库中，即使该过程并不是在统一数据库中执行的。请参见“PASSTHROUGH 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。
  - 该过程还必须存在于远程数据库中。CALL 或 EXEC 语句会被复制，但该过程包含的所有语句都不会被复制。其前提条件是在复制的数据库中该过程具有正确的效果。
- **控制语句** 在直通模式下，不复制 IF 和 LOOP 等控制语句以及任何游标操作。但会复制循环或控制结构内的所有语句。请参见“控制语句”一节 《SQL Anywhere 服务器 - SQL 的用法》。
- **游标操作** 不复制游标操作。
- **SQL SET OPTION 语句** 不复制静态嵌入式 SQL SET OPTION 语句。但会复制动态 SQL 语句。请参见“静态和动态 SQL”一节 《SQL Anywhere 服务器 - 编程》。

  例如，在直通模式下不会复制以下语句：

```
EXEC SQL SET OPTION ...
```


但会复制以下动态 SQL 语句：

```
EXEC SQL EXECUTE IMMEDIATE "SET OPTION ... "
```

- **批处理语句** 直通模式下不复制批处理语句（位于 BEGIN 和 END 之间的一组语句）。如果在直通模式下尝试使用批处理语句，则会出现错误。

另请参见

- “重新同步预订”一节第 133 页

启动和停止直通模式

使用 PASSTHROUGH 语句和 PASSTHROUGH STOP 语句启动和停止直通模式。直通会话是指在 PASSTHROUGH 语句之间输入的语句。在直通会话中输入的语句：

- 被检查是否存在语法错误。
- 被在统一数据库上执行，除非您提供 ONLY 关键字。当指定了 ONLY 时，这些语句将被发送到远程数据库，而不在统一数据库上加以执行。

  以下语句将启动一个直通会话，该会话将语句传递给一个由两个指定预订者组成的列表，而不在当前数据库中执行：

  ```
  PASSTHROUGH ONLY
  FOR userid_1, userid_2;
  ```

- 被传递给已标识的预订者数据库。直通语句会按照它们在事务日志中的记录顺序与常规的复制消息一起依次复制。
- 被在预订者数据库上执行。

定向直通语句

以下语句将启动一个直通会话，该会话将语句传递给预订 pubname 发布的所有用户：

```
PASSTHROUGH ONLY
FOR SUBSCRIPTION TO [owner].pubname statement1;
```

直通模式为累积式。以下示例中，statement_1 被发送到 user_1，而 statement_2 被发送到 user_1 和 user_2。

```
PASSTHROUGH ONLY FOR user_1 ;
statement_1;
PASSTHROUGH ONLY FOR user_2 ;
statement_2;
```

以下语句为所有远程用户停止直通会话：

```
PASSTHROUGH STOP;
```

数据修改语言 (DML)

直通模式通常用于发送数据修改语句。在这种情况下，所复制的 DML 语句在直通之前使用 *before* 模式，在直通之后使用 *after* 模式。

以下示例同时在远程数据库和统一数据库中删除同一个表。

```
-- Drop a table on the remote database
-- and at the consolidated database
PASSTHROUGH TO Joe_Remote;
DROP TABLE CrucialData;
PASSTHROUGH STOP;
```

以下示例只在远程数据库中删除一个表。

```
-- Drop a table on the remote database only
PASSTHROUGH ONLY TO Joe_Remote;
DROP TABLE CrucialData;
PASSTHROUGH STOP;
```

另请参见

- “PASSTHROUGH 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “数据修改语句”一节 《SQL Anywhere 服务器 - SQL 的用法》

重新同步预订

创建远程数据库时，可从统一数据库同时抽取模式和数据，并使用它们来建立远程数据库。此过程可以确保每个数据库都具有数据的初始副本。部署之后，您可能会在以下情况下考虑重新同步预订：

- **完成对统一数据库的重要维护之后** 例如，更改统一数据库，从而更新数据库中的每一行。缺省情况下，SQL Remote 将创建更新消息并将其发送到每个预订的远程端。这些更新消息会为每一行包括 UPDATE、DELETE 和 INSERT 语句。

  如果选择使用 SYNCHRONIZE SUBSCRIPTION 语句同步预订，则将仅发送删除预订表中所有行所需的语句，以及插入所有新行所需的 INSERT 语句。

- **当远程数据库与统一数据库不同步时** 如果远程数据库变得与统一数据库不同步，则可以尝试使用直通模式。请参见“SQL Remote 直通模式”一节第 130 页。

  如果无法使用直通模式，则可以同步预订。同步预订时，强制远程数据库与统一数据库同步。SYNCHRONIZE SUBSCRIPTION 语句包含用于删除远程数据库中预订表内容的语句，以及用于将统一数据库中预订的行插入到远程数据库中的语句。

限制

- **同步应用于整个预订** 不能同步单个表。
- **同步时数据丢失** 远程数据库上属于预订且未复制到统一数据库的所有数据都将丢失。

  同步数据库之前，使用 Sybase Central 中的 [**卸载数据库向导**] 或卸载实用程序 (dbunload) 来卸载或备份远程数据库。请参见“使用 [卸载数据库向导] 导出数据”一节 《SQL Anywhere 服务器 - SQL 的用法》和“卸载实用程序 (dbunload)”一节 《SQL Anywhere 服务器 - 数据库管理》。

同步

建议使用抽取实用程序 (dbxtract) 或 [**抽取数据库向导**] 来为指定的远程数据库抽取数据，然后手工将数据装载到远程数据库中。

> **小心**
> 运行抽取实用程序 (dbxtract) 或 [**抽取数据库向导**] 时，请勿运行消息代理 (dbremote)。

♦ 同步预订 (Sybase Central)

1. 关闭远程数据库和统一数据库上的消息代理。
2. 以具有 DBA 权限的用户身份连接到统一数据库。
3. 在左窗格中，展开 [**发布**] 目录。
4. 选择一个发布。
5. 在右窗格中，单击 [**SQL Remote 预订**] 选项卡。
6. 手工同步预订：

a. 在 [**预订者**] 列表中右击用户，然后选择 [**属性**]。

b. 单击 [**高级**] 选项卡。

c. 单击 [**立即同步**]。

单击 [**立即同步**] 按钮时，预订将受到影响。随后在属性窗口中单击 [**取消**] *不会* 取消同步操作。

7. 单击 [**确定**]。

使用消息代理 (dbremote) 同步

建议使用抽取实用程序 (dbxtract) 或 [**抽取数据库向导**] 来同步预订。请参见“同步”一节 第 133 页。

抽取大量预订或同步对经常使用的大型表的预订时，会降低数据库的访问速度。可以使用 SEND AT 子句指定当统一数据库的使用量较小时进行同步的时间。请参见“设置发送频率”一节 第 88 页。

♦ 使用消息系统 (Interactive SQL) 同步预订

1. 以具有 DBA 权限的用户身份连接到统一数据库。
2. 执行 SYNCHRONIZE SUBSCRIPTION 语句。请参见“SYNCHRONIZE SUBSCRIPTION 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。

   统一数据库上的消息代理 (dbremote) 将发送预订者的预订中所有行的一个副本。消息代理 (dbremote) 假定适当的数据库模式已在远程数据库中就位。

   预订者数据库上的消息代理 (dbremote) 接收同步消息，并用新副本 *替换* 预订表的当前内容。

小心

- **不要在远程数据库上执行 SYNCHRONIZE SUBSCRIPTION 语句** 在统一数据库上执行 SYNCHRONIZE SUBSCRIPTION 语句。
- **可能导致大量消息** 通过消息系统同步数据库需要大量的消息。另外，消息的大小可超过远程数据库的大小。通过消息链接同步多个预订会增大消息流量。

  通常，建议抽取远程数据库，然后手工装载数据。

启动预订

♦ 启动预订 (Sybase Central)

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 在左窗格中，展开 [**发布**] 目录。
3. 选择一个发布。
4. 在右窗格中，单击 [**SQL Remote 预订**] 选项卡。

5. 手工同步预订：
   a. 在 [**预订者**] 列表中右击用户，然后选择 [**属性**]。
   b. 单击 [**高级**] 选项卡。
   c. 单击 [**立即同步**]。

      单击 [**立即同步**] 按钮时，预订将受到影响。随后在属性窗口中单击 [**取消**] *不会* 取消同步操作。

♦ **启动预订 (SQL)**

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 执行 START SUBSCRIPTION 语句。请参见“START SUBSCRIPTION 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。

若要在单个事务中启动多个预订，请使用 REMOTE RESET 语句。请参见“REMOTE RESET 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。

停止预订

♦ **停止预订 (Sybase Central)**

1. 以具有 DBA 权限的用户身份连接到数据库。
2. 在左窗格中，展开 [**发布**] 目录。
3. 选择所需的发布。
4. 在右窗格中，单击 [**SQL Remote 预订**] 选项卡。
5. 要手工同步预订，请在 [预订者] 列表中选择用户，然后选择 [**属性**]。

   单击 [**高级**] 选项卡。在此选项卡上，单击 [**立即停止**] 以停止预订。

   单击 [**立即停止**] 按钮时，预订将受到影响。随后在属性窗口上单击 [**取消**] *不会* 取消您的停止同步操作。

SQL Remote 参考

本节将提供 SQL Remote 的参考资料。

SQL Remote 实用程序和选项参考

目录

消息代理 (dbremote)

作用

发送和应用 SQL Remote 消息并维护消息跟踪系统以确保消息的传送。

语法

dbremote [*options*] [*directory*]

选项

选项	说明

@data	此选项用于从指定的环境变量或配置文件中读入选项。如果存在具有相同名称的环境变量和配置文件，则使用环境变量。请参见“使用配置文件”一节 《SQL Anywhere 服务器 - 数据库管理》。
	
	如果要保护口令或配置文件中的其它信息，可以使用文件隐藏实用程序对配置文件的内容进行模糊处理。请参见“文件隐藏实用程序 (dbfhide)”一节 《SQL Anywhere 服务器 - 数据库管理》。
	
	环境变量可以包含任何一组选项。例如，以下两条语句的第一句设置一个环境变量，该变量包含了一组 SQL Remote 进程选项，该进程启动时高速缓存大小为 4 MB，只接收消息，并且连接到名为 **myserver** 的数据库服务器上的名为 **field** 的数据库。SET 语句应全部在一行上输入：
	
	`SET envvar=-m 4096 -r -c "ENG=myserver;DBN=field;UID=sa;PWD=sysadmin" dbremote @envvar`
	
	配置文件包含换行符，并且可以包含任何一组选项。例如，以下命令文件包含了消息代理的一组选项，该消息代理在启动时高速缓存大小为 4 MB，仅发送消息，并连接到名为 **myserver** 的数据库服务器上的名为 **field** 的数据库：
	
	`-m 4096`
	`-s`
	`-c "ENG=myserver;DBN=field;UID=sa;PWD=sysadmin"`
	
	如果此配置文件保存为 *c:\config.txt*，则在命令中可按如下方式使用：
	
	`dbremote @c:\config.txt`

选项	说明
-a	处理收到的消息（收件箱中的那些消息），但不将这些消息应用于数据库。此选项与 -v（用于详细输出）和 -p（不清除消息）一起使用，可以帮助找出与进来的消息有关的问题。如果不与 -p 一起使用，此选项会清除收件箱且不应用消息，这在重新开始预订的情况下可能会非常有用。
-b	在批处理模式下运行。在此模式下，消息代理处理进来的消息，扫描一次事务日志并处理外发的消息，然后停止。
-c *"keyword=value; ..."*	指定连接参数。如果未指定此选项，则使用环境变量 SQLCONNECT。 例如，以下语句在位于 *c:\mydata.db* 中的数据库文件中运行 dbremote，并使用用户 ID **DBA** 和口令 **sql** 进行连接： `dbremote -c "UID=DBA;PWD=sql;DBF=c:\mydata.db"` 消息代理必须由具有 REMOTE DBA 权限或 DBA 权限的用户运行。请参见“授予 REMOTE DBA 权限”一节第 30 页。 消息代理支持全部的 SQL Anywhere 连接参数。请参见“连接参数”一节 《SQL Anywhere 服务器 - 数据库管理》。
-dl	在消息代理窗口中或命令提示符处显示消息；如果指定，还在日志文件中显示消息。
-ek *key*	指定需要在命令提示符处被系统提示输入加密密钥以用于高度加密的数据库。如果您有一个高度加密的数据库，则必须提供加密密钥，才能使用数据库或事务日志，包括脱机事务日志。对于高度加密数据库，必须指定 -ek 或 -ep，但不要同时指定这两者。如果不为高度加密的数据库指定密钥，该命令将失败。
-ep	指定需要系统提示输入加密密钥。使用此选项将显示一个窗口，可以在其中输入加密密钥。这样，加密密钥从不以明文显示，提供了额外的安全保证。对于高度加密数据库，必须指定 -ek 或 -ep，但不要同时指定这两者。如果不为高度加密的数据库指定密钥，该命令将失败。
-g *n*	指示消息代理将所含操作少于 *n* 的事务与后面的事务组合在一起。缺省值是 20 个操作。增加 *n* 的值可以通过执行较少的提交来加快进来的消息的处理。但是，增大事务也会导致死锁和阻塞。

选项	说明
-l *length*	指定要发送的每条消息的最大长度（以字节为单位）。较长的事务被拆分成多个消息。缺省值是 50000 个字节，最小长度是 10000。 **小心** 在安装中，最大消息长度必须在所有站点都是相同的。 对于具有有限内存分配的平台，该值必须小于操作系统的最大内存分配。
-m *size*	指定可供消息代理生成消息以及将进来的消息存入高速缓存之用的最大内存量。可将允许的大小指定为 *n*（以字节为单位）、*n*K 或 *n*M。缺省值为 2048 KB (2 MB)。 当所有远程数据库接收的是所复制操作的唯一子集时，将同时为每一远程数据库分别生成不同的消息。为接收相同操作的一组远程用户只生成一个消息。如果所使用的内存超出 -m 值，则消息会在达到其最大大小（由 -l 选项指定）前发送出去。 在消息到达后，这些消息会在应用前由消息代理存储在内存中。将消息存入高速缓存中可以防止重新从消息系统读取顺序不对的消息，以免在大型系统中降低性能。超过了 -m 选项指定的内存使用量后，将按使用频率由低到高的顺序刷新消息。
-ml *directory*	指定脱机事务日志镜像文件的位置。此选项使 dbremote 能够在出现以下两种情形中的任意一种时，删除旧的事务日志镜像文件： ● 脱机事务日志镜像与事务日志镜像不在同一目录中 ● dbremote 与远程数据库服务器不在同一台计算机上运行 在典型设置中，活动事务日志镜像与重命名的事务日志镜像位于同一目录中，并且 dbremote 与远程数据库在同一台计算机上运行，因此不需要此选项，旧的事务日志镜像文件也会被自动删除。只有在将 delete_old_logs 数据库选项设置为 Off 之外的其它值时，此目录中的事务日志才会受到影响。
-o *file*	将消息打印输出到输出日志文件。缺省情况下这些消息将输出到屏幕。

选项	说明
-os *size*	指定用于记录输出消息的最大文件大小。允许的大小可以指定为 *n*（字节）、*n***K** (KB) 或 *n***M** (MB)。缺省情况下，没有上限，而最小限制为 10000 个字节。 在 SQL Remote 将输出消息记录到输出日志文件之前，它会首先检查当前文件大小。如果日志消息使文件大小超出了指定大小，则 SQL Remote 会将输出文件重命名为 *yymmddxx.dbr*，其中 *xx* 是从 AA 到 ZZ 的顺序字符，*yymmdd* 表示当前的年月日。 如果消息代理长期在连续模式下运行，则可以通过该选项手动删除旧的输出日志文件并释放磁盘空间。
-ot *file*	截断输出日志文件，然后将输出消息追加到该文件。缺省情况下这些消息将输出到屏幕。
-p	不清除消息。
-q	以最小化窗口启动消息代理。此选项仅适用于 Windows 操作系统。
-qc	完成时关闭 SQL Remote 窗口。
-r	接收消息。如果 -r 和 -s 均未指定，消息代理将执行全部两个阶段。否则，将仅执行指定的阶段。 如果用 -r 选项运行消息代理，消息代理将在连续模式下运行。要使消息代理在收到消息后关闭，请同时使用 -b 选项和 -r 选项。
-rd *minutes*	指定对进来的消息的轮询频率。缺省情况下，消息代理每分钟轮询进来的消息。通过此选项（rd 代表 **receive delay**）可配置轮询频率，这在轮询开销非常大时会很有用。 您可以在数字后面加上 **s** 后缀来指示秒数，如果要频繁进行轮询，这会十分有用。例如，以下命令每三十秒轮询一次。 `dbremote -rd 30s` -rd 选项常与 -rp 选项一起使用，-rp 选项用于设置消息代理请求重发丢失的消息前所等待的轮询数。 请参见“在接收消息时提高性能”一节第 92 页。
-ro *filename*	将远程输出记录到文件。此选项用在统一站点。当远程数据库被配置为向统一数据库发送输出日志信息时，该选项将这些信息写入一个文件中。此选项可以帮助管理员解决远程站点中的问题。 请参见“从远程数据库收集错误”一节第 124 页。

选项	说明
-rp *number*	指定在认为消息丢失之前接收的轮询数目。在以连续模式运行时，消息代理按指定的时间间隔轮询消息。在轮询指定次数后（缺省情况下是一次），如果发现缺少某一消息，则消息代理假设该消息已丢失并请求重新发送该消息。在较慢的消息系统上，该行为可能会导致许多不必要的重新发送请求。可以使用此选项来设置在发出重新发送请求之前的轮询次数，以将重新发送请求数降至最低。 有关配置此选项的详细信息，请参见“在接收消息时提高性能”一节第 92 页。 -rp 选项常与用于设置进来消息轮询频率的 -rd 选项一起使用。
-rt *filename*	启动时截断输出日志文件，然后将日志输出从远程数据库追加到该文件。此选项用在统一站点。除了文件会在启动时被截断以外，它与 -ro 选项相同。
-ru *time*	指定收到重新发送请求后等待重新扫描日志的时间。 控制 **resend urgency**。这是检测到重新发送请求和消息代理开始实现该请求之间的时间。使用此选项可以帮助消息代理在重新扫描日志前收集来自多个用户的重新发送请求。时间单位可以是 s（秒）、m（分钟）、h（小时）或 d（天）。
-s	发送消息。如果 -r 和 -s 均未指定，消息代理将执行全部两个阶段。否则，将仅执行指定的阶段。
-sd *time*	控制数据库事务日志各轮询间的延迟。只有在连续模式下运行时，才会使用 -sd 选项。 控制 **send delay**，即对于要发送的更多事务日志数据而进行的各轮询之间的间隔时间。
-t	复制所有触发器。如果使用此选项，必须确保在远程数据库中触发器操作不会执行两次（一次由在远程站点被触发的触发器执行，一次由从统一数据库复制操作的显式应用执行）。 要确保触发操作不会执行两次，可以使用 IF CURRENT REMOTE USER IS NULL ...END IF 语句包裹触发器正文。 “使用 CURRENT REMOTE USER 特殊常量”一节第 47 页.
-u	仅处理脱机事务日志中的事务。该选项防止消息代理处理最后一次备份之后的事务。使用此选项后，如果脱机事务日志中没有外发的事务以及进来事务的确认，就不能将它们发送出去。 这意味着只处理来自重命名日志的事务。

选项	说明
-ud	在 Unix 平台上，将消息代理作为守护程序运行。如果将消息代理作为守护程序运行，则还必须提供 -o 或 -ot 选项，以记录输出信息。 如果将消息代理作为守护程序运行并正在使用 FTP 或 SMTP 消息链接，则必须将消息链接参数存储到数据库中，因为消息代理在作为守护程序运行时不会提示用户提供这些选项。 有关消息链接参数的信息，请参见“设置远程消息类型控制参数”一节第 107 页。
-ux	在 Solaris 和 Linux 上，打开 SQL Remote 消息窗口。 当指定 -ux 时，dbremote 必须能够找到一个可用显示。如果找不到（例如，因为未设置 DISPLAY 环境变量或 X 窗口服务器未运行），则 dbremote 将无法启动。在 Windows 上，SQL Remote 消息窗口会自动显示。
-v	显示详细输出。此选项会将消息中包含的 SQL 语句显示到消息窗口；此外，如果使用了 -o 或 -ot 选项，则还将这些 SQL 语句输出到日志文件中。
-w *n*	指定要应用进来的消息的数据库工作线程数。在 Windows Mobile 中不支持此选项。 缺省值为零，这意味着所有消息都由（并且仅由）主线程应用。如果该值为 1，将有一个用于从消息系统接收消息的线程和一个用于将消息应用于数据库的线程。数据库工作线程数上限为 50。 利用 -w 选项，能够通过硬件升级来增加进来的消息的吞吐量。通过将统一数据库放置到可以执行多个并发操作的设备上（分条的逻辑驱动器的 RAID 阵列），可以提高进来的消息的吞吐量。如果在运行消息代理的计算机中使用多个处理器，也可以提高进来的消息的吞吐量。 对于不能执行多个并发操作的硬件，-w 选项不会显著改进性能。 进来的来自一个远程数据库的消息永远也不会应用于多个线程上。来自一个远程数据库的消息将始终以正确的顺序依次加以应用。

选项	说明
-x [*size*]	在扫描事务日志寻找外传消息后，重命名并重新启动该事务日志。在某些情况下，将数据复制到统一数据库可以取代备份远程数据库，或者取代在数据库服务器关闭后重命名事务日志。 如果提供可选的 *size* 限定符，则仅当事务日志超过指定的大小后才将其重命名。可将允许的大小指定为 *n*（以字节为单位），*n*K 或 *n*M。缺省值为 0。
Directory	指定保留旧事务日志的目录。 *directory* 参数是可选的，用于指定保存旧事务日志的目录，以便消息代理能够在启动当前日志前有权访问事件。

说明

消息代理发送和应用用于 SQL Remote 复制的消息，并且维护消息跟踪系统以确保消息的传送。

消息代理可执行文件的名称为 dbremote。

您还可以通过调用 DBTools 库，从您自己的应用程序运行消息代理。有关详细信息，请参见位于 SQL Remote 安装目录的 *h* 子目录中的 *dbrmt.h* 文件。

消息代理命令中的用户 ID 必须具有 REMOTE DBA 或 DBA 权限。

消息代理使用多个数据库连接。请参见“消息代理 (dbremote)”一节第 140 页。

有关 REMOTE DBA 权限的信息，请参见“授予 REMOTE DBA 权限”一节第 30 页。

消息系统控制参数

SQL Remote 使用若干注册表设置来控制消息链接行为的各方面。

消息链接控制参数存储在以下位置：

- **Windows** 存储在注册表的以下位置中：

```
\\HKEY_CURRENT_USER
 \Software
  \Sybase
   \SQL Remote
```

有关注册表设置的列表，请参见“SQL Remote 消息系统”一节第 105 页中与各消息系统相对应的部分。

抽取实用程序 (dbxtract)

从 SQL Anywhere 统一数据库中抽取远程数据库。

语法

dbxtract [*options*] [*directory*] *subscriber*

选项	说明
@data	从配置文件读入选项。请参见“@data 服务器选项”一节 《SQL Anywhere 服务器 - 数据库管理》。 此选项用于从指定的环境变量或配置文件中读入选项。如果存在具有相同名称的环境变量和配置文件，则使用环境变量。请参见“使用配置文件”一节 《SQL Anywhere 服务器 - 数据库管理》。 如果要保护口令或配置文件中的其它信息，可以使用文件隐藏实用程序对配置文件的内容进行模糊处理。请参见“文件隐藏实用程序 (dbfhide)”一节 《SQL Anywhere 服务器 - 数据库管理》。
-ac *"keyword=value; ..."*	连接到在连接字符串中指定的数据库以进行重装。 您可以使用此选项，将数据库卸载操作与将结果重装入现有数据库的操作合并执行。 例如，以下命令（所有内容应在一行中输入）将预订者 field_user 的数据副本装载到名为 *c:\field.db* 的现有数据库文件中： `dbxtract -c "UID=DBA;PWD=sql;DBF=c:\cons.db" -ac "UID=DBA;PWD=sql;DBF=c:\field.db" field_user` 如果使用此选项，则不会在磁盘上创建数据的副本，因而无需在命令中指定卸载目录。这可以使数据更安全，但对性能会造成一定的影响。
-al *filename*	如果使用 -an 选项，则指定新数据库的事务日志文件名。

选项	说明
-an *database*	创建与所抽取的数据库具有相同设置的数据库文件，并自动重装。 您可以使用此选项，将卸载数据库、创建新数据库和装载数据这三个操作组合起来。 例如，以下命令（所有内容应在一行中输入）将创建一个名为 *c:\field.db* 的新数据库文件，并将 *c:\cons.db* 的 field_user 预订者的模式和数据复制到其中： `dbxtract -c "UID=DBA;PWD=sql;DBF=c:\cons.db" -an c:\field.db field_user` 如果使用此选项，则不会在磁盘上创建数据的副本，因而无需在命令中指定卸载目录。这可以使数据更安全，但对性能会造成一定的影响。
-ap *size*	设置新数据库的页面大小。如果使用了 -an，则忽略此选项。数据库的页面大小（以字节为单位）可以是 2048、4096、8192、16384 或 32768，缺省值为原数据库的页面大小。如果数据库服务器上已有数据库在运行，则服务器的页面大小（使用 -gp 选项设置）必须大到足以处理新的页面大小。请参见“-gp 服务器选项”一节《SQL Anywhere 服务器 - 数据库管理》。
-b	不启动预订。如果指定了此选项，则必须使用 START SUBSCRIPTION 语句显式启动位于统一数据库的预订（用于远程数据库）和位于远程数据库的预订（用于统一数据库），以便开始复制。请参见“START SUBSCRIPTION 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》。
-c *"keyword=value; ..."*	提供字符串形式的数据库连接参数： 用户 ID 应具有 DBA 权限以确保用户有权访问数据库中的所有表。 例如，以下语句（所有内容应在一行中键入）以用户 ID DBA 和口令 sql 进行连接，从正在 sample_server 数据库服务器上运行的示例数据库中为远程用户 ID joe_remote 抽取数据库。这些数据被卸载到 *c:\extract* 目录中。 `dbxtract -c "ENG=sample_server;DBN=demo; UID=DBA;PWD=sql" c:\extract joe_remote` 如果未指定连接参数，则使用来自 SQLCONNECT 环境变量的连接参数（如果已设置）。
-d	仅抽取数据。如果指定了此选项，则不卸载模式定义且不在远程数据库中创建发布和预订。此选项适用于：已存在具有正确模式的远程数据库且该数据库只需用数据来填充。

选项	说明
-ea *alg*	指定新数据库的加密算法此选项允许您选择使用何种高度加密算法来加密您的新数据库。您可以选择 AES（缺省值）或 FIPS 认可的算法 AES_FIPS。AES_FIPS 使用独立的库并且与 AES 不兼容。 为获得更高的安全性，可指定 AES 或 AES256 以分别实现 128 位或 256 位高度加密。指定 AES_FIPS 或 AES256_FIPS 以分别实现 128 位或 256 位 FIPS 认可的加密。对于高度加密，还必须指定 -ek 或 -ep 选项。有关高度加密的详细信息，请参见“高度加密”一节 《SQL Anywhere 服务器 - 数据库管理》。 要创建未加密的数据库，请指定 -ea none 或者不包含 -ea 选项（并且不要指定 -e、-et、-ep 或 -et）。 如果未指定 -ea 选项，则缺省行为如下： ● 如果未指定 -ek、-ep 或 -et，则为 -ea none ● 如果指定了 -ek 或 -ep（带或不带 -et），则为 -ea AES ● 如果指定了 -et，而未指定 -ek 或 -ep，则为 -ea simple 算法名称是不区分大小写的。 **需要单独授予许可的组成部分** ECC 加密和 FIPS 认证的加密需要单独的许可。所有高度加密技术受出口法规约束。 请参见“单独授权的组件”一节 《SQL Anywhere 11 - 简介》。
-ek *key*	指定新数据库的加密密钥。使用此选项，您可以通过直接在命令中指定加密密钥，来创建高度加密的数据库。用于加密数据库的算法是 AES 还是 AES_FIPS，由 -ea 选项指定。如果您指定了 -ek 选项而没有指定 -ea，则使用 AES 算法。 **小心** 对于高度加密的数据库，请务必将密钥的副本保存在安全的位置。如果丢失了加密密钥，则无法访问数据，即使有技术支持人员的协助也是如此。此时必须放弃该数据库并创建一个新的数据库。
-ep	提示输入新数据库的加密密钥。此选项指定您希望通过在窗口中键入加密密钥来创建高度加密的数据库。这样，加密密钥决不会以明文显示，从而提供了额外的安全保证。 加密密钥必须输入两次，以确认密钥的输入是正确的。如果密钥不匹配，则初始化失败。请参见“高度加密”一节 《SQL Anywhere 服务器 - 数据库管理》。

选项	说明
-er	在卸载过程中，从加密表中删除加密。 从启用了表加密的数据库进行抽取时，必须指定 -er 或 -et 来指示新数据库是否启用表加密，否则在尝试向该新数据库装载数据时会发生错误。 以下命令将拥有加密表的数据库 (*cons.db*) 抽取到不启用表加密的新数据库 (*field.db*) 中，即解除了所有加密表的加密： `dbxtract -an c:\field.db -er -c "UID=DBA;PWD=sql;DBF=c:\cons.db;DBKEY=29bN8cj1z field_user"`
-et	在新数据库中启用数据库表加密（还必须指定 -an 或 -ar）。如果指定 -et 选项但未指定 -ea 选项，将会使用 AES 算法。如果指定 -et 选项，就必须也指定 -ep 或 -ek。可以将新数据库的表加密设置更改为与所卸载的数据库不同的设置。 在重建已启用表加密的数据库时，必须指定 -er 或 -et 来指明新数据库是否启用表加密，否则在尝试向该新数据库装载数据时可能会出错。 下面的示例将使用简单加密算法进行表加密的数据库 (*cons.db*) 卸载到启用表加密的新数据库 (*field.db*) 中，并采用 AES_FIPS 加密算法，密钥为 34jh： `dbxtract -an c:\field.db -et -ea AES_FIPS -ek 34jh -c "UID=DBA;PWD=sql;DBF=c:\cons.db field_user"`
-f	抽取完全限定的发布。在大多数情况下，您不需要为远程数据库抽取完全限定的发布定义，因为它通常会将所有行复制回统一数据库。 但是，对于多层设置，或者对于在其中远程数据库所具有的行不位于统一数据库中的设置，您可能需要完全限定的发布。
-ii	执行内部卸载和内部重装。通过使用此选项，可以强制重装脚本使用内部 UNLOAD 和 LOAD TABLE 语句（而非 Interactive SQL OUTPUT 和 INPUT 语句）来分别卸载和装载数据。这一操作组合是缺省行为。 外部操作采用相对于 dbxtract 的当前工作目录的数据文件路径，而内部语句则采用相对于数据库服务器的路径。
-ix	执行内部卸载和外部重装。通过使用此选项，可强制重装脚本使用内部 UNLOAD 语句卸载数据，并使用 Interactive SQL INPUT 语句将数据装载到新数据库中。 外部操作采用相对于 dbxtract 的当前工作目录的数据文件路径，而内部语句则采用相对于数据库服务器的路径。

选项	说明
-l *level*	以指定隔离级别执行所有抽取操作缺省设置是隔离级别 0。如果从活动数据库服务器抽取数据库，请选择在隔离级别 3 运行该抽取操作，以确保抽取的数据库中的数据与数据库服务器上的数据保持一致。提高隔离级别可能导致抽取实用程序 (dbxtract) 使用大量锁定，并且可能限制其他用户使用数据库。请参见“抽取实用程序 (dbxtract)”一节第 147 页。
-n	仅抽取模式定义。使用此定义，将不卸载任何数据。重装文件只包含用来建立数据库模式的 SQL 语句。您可以使用 SYNCHRONIZE SUBSCRIPTION 语句通过消息系统装载数据。发布、预订、PUBLISH 和 SUBSCRIBE 权限都是模式的一部分。 `dbxtract -c "UID=DBA;PWD=sql;DBF=c:\remote\cons\cons.db" -n "c:\remote\reload.sql" UserName`
-nl	抽取结构（与 -n 选项的行为相同），但结果 *reload.sql* 文件中还包含每个表的 LOAD TABLE 或 INPUT 语句。使用此选项时不会抽取任何用户数据。如果指定 -nl，还必须包括一个数据目录，以便可以生成 LOAD/INPUT 语句，即使没有文件写入该目录中。此选项使您可以在不卸载数据的情况下生成重装脚本。可以通过指定 -d 来抽取数据。如果数据库中的某个表包含不应卸载的数据，可以使用 `dbxtract -d -e table-name` 避免卸载该表中的数据。
-o *file*	将消息输出到输出日志文件。
-p *character*	指定转义字符。通过此选项可以使用其它字符替代缺省的转义字符 (\)。
-q	安静地运行：不显示消息，也不显示窗口。如果指定此选项，就必须也指定 -y，否则操作将失败。 此选项只用于命令行实用程序。
-r *file*	指定生成的重装 Interactive SQL 命令文件的名称。 恢复命令文件的缺省名称为当前目录中的 *reload.sql*。可使用此选项指定不同的文件名。
-u	在卸载操作期间不排序数据。缺省情况下，每个表中的数据都按主键排序。使用 -u 选项可以加快卸载，但会降低向远程数据库装载数据的速度。
-v	显示详细消息。所卸载的表的名称、卸载行数，以及所使用的 SELECT 语句。

选项	说明
-xf	排除外键。如果远程数据库包含统一数据库模式的子集，并且在远程数据库中不存在某些外键引用，则可以使用此项。
-xh	排除过程挂接。
-xi	执行外部卸载和内部重装。卸载数据库的缺省行为是使用 UNLOAD 语句，这是由数据库服务器执行的。如果选择外部卸载，则 dbxtract 会改为使用 OUTPUT 语句。在客户端上执行 OUTPUT 语句。 外部操作采用相对于 dbxtract 的当前工作目录的数据文件路径，而内部语句则采用相对于数据库服务器的路径。
-xp	不从数据库抽取存储过程。
-xt	不从数据库抽取触发器。
-xv	不从数据库抽取视图。
-xx	执行外部卸载和外部装载。使用 OUTPUT 语句卸载数据，使用 INPUT 语句将数据装入新的数据库中。 缺省的卸载行为是使用 UNLOAD 语句，缺省的装载行为是使用 LOAD TABLE 语句。内部 UNLOAD 和 LOAD TABLE 语句的执行速度要快于 OUTPUT 和 INPUT。 外部操作采用相对于 dbxtract 的当前工作目录的数据文件路径，而内部语句则采用相对于数据库服务器的路径。
-y	覆盖命令文件，无须确认。如果不指定此选项，将提示确认覆盖现有命令文件。
directory	指定写入文件的目录。指定 -an 或 -ac 时不需要此选项。
subscriber	指定正在为其抽取数据库的预订者。

注释

缺省情况下，抽取实用程序 (dbxtract) 在隔离级别 0 运行。如果从活动数据库服务器抽取数据库，请选择在隔离级别 3 运行该抽取操作，以确保抽取的数据库中的数据与数据库服务器上的数据保持一致。在隔离级别 3 运行可能会由于需要大量锁定而延长数据库服务器上其它进程的周转时间。建议您在服务器不繁忙时运行抽取实用程序 (dbxtract)，或者对数据库的副本运行该抽取实用程序。

抽取实用程序 (dbxtract) 会创建一个命令文件和一组关联的数据文件。该命令文件可在新初始化的数据库上运行，以创建数据库对象及装载远程数据库的数据。

缺省情况下，命令文件被命名为 *reload.sql*。

如果远程用户为一个组，则抽取属于该组的所有用户 ID。这允许远程数据库上存在具有不同用户 ID 的多个用户，并且不需要自定义抽取进程。

在将抽取实用程序 (dbxtract) 或 [**抽取数据库向导**] 用于版本 10.0.0 或更高版本的数据库时，所使用的 dbxtract 的版本必须与用于访问该数据库的数据库服务器的版本相符。如果将较旧版本的 dbxtract 和相对较新版本的数据库服务器一起使用，会报告出现错误，反之亦然。

抽取实用程序 (dbxtract) 和 [**抽取数据库向导**] 不会卸载在数据库创建期间为 **dbo** 用户 ID 所创建的对象。在数据被卸载后将失去对这些对象进行的更改（例如，重新定义系统过程）。而在数据库初始化后由 **dbo** 用户 ID 所创建的所有对象都会被抽取实用程序 (dbxtract) 卸载，因此这些对象会保留下来。

另请参见

- “抽取远程数据库”一节第 79 页
- “抽取数据”一节 《SQL Anywhere 服务器 - SQL 的用法》

SQL Remote 选项

功能

复制选项是数据库选项，用于对复制行为加以控制。

语法

SET [**TEMPORARY**] **OPTION**
[*userid*. | **PUBLIC**.]*option-name* = [*option-value*]

参数

参数	说明
option-name	所更改的选项的名称。
option-value	包含用于该选项的设置的字符串。

说明

这些选项由消息代理使用，并且应针对在消息代理命令中指定的用户 ID 来设置这些选项。还可以设置它们以用于常规公共用途。

提供以下选项。

选项	值	缺省
“blob_threshold 选项 [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》	整数（以字节为单位）	256
“compression 选项 [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》	从 -1 到 9 之间的整数	6
“delete_old_logs 选项 [MobiLink 客户端] [SQL Remote] [复制代理]”一节 《SQL Anywhere 服务器 - 数据库管理》	On、Off、Delay、*n* 天	Off
“external_remote_options [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》	On、Off	Off
“qualify_owners 选项 [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》	On、Off	On

选项	值	缺省
“quote_all_identifiers 选项 [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》	On、Off	Off
“replication_error 选项 [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》	存储过程名	（无过程）
“replication_error_piece 选项 [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》	存储过程名	（无过程）
“save_remote_passwords 选项 [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》	On、Off	On
“sr_date_format 选项 [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》	*日期字符串*	*YYYY/MM/DD*
“sr_time_format 选项 [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》	*时间字符串*	*HH:NN:SS.SSSSSS*
“sr_timestamp_format [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》	*时间戳字符串*	*YYYY/MM/DD HH:NN:SS.SSSSSS*
“subscribe_by_remote 选项 [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》	On、Off	On
“verify_all_columns 选项 [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》	On、Off	Off
“verify_threshold 选项 [SQL Remote]”一节 《SQL Anywhere 服务器 - 数据库管理》	整数（以字节为单位）	1000

SQL Remote 系统过程

以下存储过程名称和参数提供用于在 SQL Remote 数据库自定义复制的接口。

注意

除非另有规定，否则以下条件适用于事件挂接过程：

- 存储过程必须具有 DBA 权限。
- 过程不得提交或回退操作，也不得执行任何执行隐式提交的操作。该过程的操作通过调用应用程序自动提交。
- 您可以通过启用消息代理详细模式，排除挂接问题。

#hook_dict 表

调用挂接前用以下 CREATE 语句创建 #hook_dict 表：

```
CREATE TABLE #hook_dict(
NAME VARCHAR(128) NOT NULL UNIQUE,
value VARCHAR(255) NOT NULL );
```

消息代理使用 #hook_dict 表将值传送给挂接函数；挂接函数使用 #hook_dict 表将值传送回消息代理。

sp_hook_dbremote_begin 系统过程

在复制进程开始时使用此系统过程添加自定义动作。

#hook_dict 表中各行

名称	值	说明
send	**true** 或者 **false**	指示进程是否正在执行复制的发送阶段。
receive	**true** 或者 **false**	指示进程是否正在执行复制的接收阶段。

说明

如果存在此名称的过程，则在消息代理启动时调用它。

sp_hook_dbremote_end 系统过程

使用此系统过程在事件代理刚好退出之前添加自定义动作。

#hook_dict 表中各行

名称	值	说明
send	**true** 或者 **false**	指示进程是否正在执行复制的发送阶段。
receive	**true** 或者 **false**	指示进程是否正在执行复制的接收阶段。
exit code	**integer**	非零退出代码指示错误。

说明

如果存在此名称的过程，则在消息代理关闭前将该过程作为最后的事件调用。

sp_hook_dbremote_shutdown 系统过程

使用此系统过程使消息代理关闭。

#hook_dict 表中各行

名称	值	说明
send	**true** 或者 **false**	指示进程是否正在执行复制的发送阶段。
receive	**true** 或者 **false**	指示进程是否正在执行复制的接收阶段。
shutdown	**true** 或者 **false**	在调用过程时此行是 **false**。如果过程将该行更新为 **true**，则消息代理将关闭。

说明

如果存在此名称的过程，则在消息代理既没有发送消息也没有接收消息时调用该过程，并且允许消息代理的挂接启动关闭。

sp_hook_dbremote_receive_begin 系统过程

使用此系统过程在复制的接收阶段开始前执行动作。

#hook_dict 中各行

无

sp_hook_dbremote_receive_end 系统过程

使用此系统过程在复制的接收阶段结束后执行动作。

#hook_dict 中各行

无

sp_hook_dbremote_send_begin

使用此存储过程在复制的发送阶段开始前执行动作。

#hook_dict 中各行

无

sp_hook_dbremote_send_end

使用此存储过程在复制的发送阶段结束后执行动作。

#hook_dict 中各行

无

sp_hook_dbremote_message_sent

使用此存储过程在发送任何消息后执行动作。

#hook_dict 中各行

名称	值
remote user	消息目标。

sp_hook_dbremote_message_missing

使用此存储过程在消息代理已确定远程用户缺少一个或多个消息时执行动作。

#hook_dict 中各行

名称	值
remote user	需要重新发送消息的远程用户的名称。

sp_hook_dbremote_message_apply_begin

使用此存储过程在消息代理应用来自用户的一组消息之前执行动作。

#hook_dict 中各行

名称	值
remote user	发送将要应用的消息的远程用户的名称。

sp_hook_dbremote_message_apply_end

使用此存储过程在消息代理紧接着应用来自用户的一组消息之后执行动作。

#hook_dict 中各行

名称	值
remote user	发送已应用的消息的远程用户的名称。

SQL Remote 系统对象

目录

SQL Remote 系统表

SQL Remote 系统信息保存在 SQL Anywhere 目录中。更详细的信息保存在一组系统视图中。以下是可用于访问 SQL Remote 数据的视图：

- “SYSARTICLE 系统视图”一节 《SQL Anywhere 服务器 - SQL 参考》
- “SYSARTICLECOL 系统视图”一节 《SQL Anywhere 服务器 - SQL 参考》
- “SYSPUBLICATION 系统视图”一节 《SQL Anywhere 服务器 - SQL 参考》
- “SYSREMOTEOPTION 系统视图”一节 《SQL Anywhere 服务器 - SQL 参考》
- “SYSREMOTEOPTIONTYPE 系统视图”一节 《SQL Anywhere 服务器 - SQL 参考》
- “SYSREMOTETYPE 系统视图”一节 《SQL Anywhere 服务器 - SQL 参考》
- “SYSREMOTEUSER 系统视图”一节 《SQL Anywhere 服务器 - SQL 参考》
- “SYSSUBSCRIPTION 系统视图”一节 《SQL Anywhere 服务器 - SQL 参考》

SQL Remote 的 SQL 语句

目录

SQL Remote 语句

以下是用于执行 SQL Remote 命令的 SQL 语句：

- “ALTER PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “ALTER REMOTE MESSAGE TYPE 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “CREATE PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “CREATE REMOTE MESSAGE TYPE 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “CREATE SUBSCRIPTION 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “CREATE TRIGGER 语句”一节 《SQL Anywhere 服务器 - SQL 参考》
- “DROP PUBLICATION 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “DROP REMOTE MESSAGE TYPE 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “DROP SUBSCRIPTION 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “GRANT CONSOLIDATE 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “GRANT PUBLISH 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “GRANT REMOTE 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “GRANT REMOTE DBA 语句 [MobiLink] [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “PASSTHROUGH 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “REMOTE RESET 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “REVOKE CONSOLIDATE 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “REVOKE PUBLISH 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “REVOKE REMOTE 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “REVOKE REMOTE DBA 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “SET REMOTE OPTION 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “START SUBSCRIPTION 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “STOP SUBSCRIPTION 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “SYNCHRONIZE SUBSCRIPTION 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》
- “UPDATE 语句 [SQL Remote]”一节 《SQL Anywhere 服务器 - SQL 参考》

术语表

术语表

Adaptive Server Anywhere (ASA)

SQL Anywhere Studio 的关系数据库服务器组件，专供在移动和嵌入式环境中使用，或作为中小型企业的服务器使用。在版本 10.0.0 中，Adaptive Server Anywhere 更名为 SQL Anywhere 服务器，SQL Anywhere Studio 更名为 SQL Anywhere。

另请参见： “SQL Anywhere”一节第 184 页

包

Java 中相关类的集合。

被引用对象

一种对象（如表），该对象在另一个对象（如视图）的定义中被直接引用。

另请参见： “主键”一节第 194 页

编码

也称作字符编码，编码是一种方法，通过该方法可以将字符集中的每个字符映射到一个或多个字节的信息，这些信息通常以十六进制数字表示。编码的一个例子是 UTF-8。

另请参见：

- “字符集”一节第 194 页
- “代码页”一节第 169 页
- “归类”一节第 173 页

标识符

用于引用数据库对象（如表或列）的字符串。标识符可以包含 A 到 Z、a 到 z、0 到 9、下划线 (_)、at 符号 (@)、数字符号 (#) 或美元符号 ($) 中的任何字符。

并发

同时执行两个或更多个独立并且可能存在竞争关系的进程。SQL Anywhere 会自动使用锁定来隔离事务，并确保每个并发应用程序看到的数据集均一致。

另请参见：

- “事务”一节第 181 页
- “隔离级别”一节第 172 页

参考数据库

MobiLink 中一种用于 UltraLite 客户端开发的 SQL Anywhere 数据库。在开发过程中，可以将一个 SQL Anywhere 数据库同时作为参考数据库和统一数据库使用。通过其它产品建立的数据库无法用作参考数据库。

参照完整性

遵守数据一致性控制规则（具体而言，不同表中主键值与外键值之间的关系）。若要实现参照完整性，每个外键中的值必须与被引用表中行的主键值相符。

另请参见：

- “主键”一节第 194 页
- “外键”一节第 186 页

策略

QAnywhere 中指定应在何时进行消息传输的方式。

插件模块

Sybase Central 中一种用于访问和管理产品的方法。当您安装相应的产品时，插件通常会自动安装并注册 Sybase Central。通常，插件在 Sybase Central 主窗口中作为顶级容器出现，并且使用产品本身的名称，如 SQL Anywhere。

另请参见： “Sybase Central”一节第 185 页

查询

一条或一组 SQL 语句，用于访问和/或操作数据库中的数据。

另请参见： “SQL”一节第 184 页

冲突解决

在 MobiLink 中，冲突解决是指一种逻辑，它指定当两个用户修改不同远程数据库上同一行时的处理方法。

重定向器

一种 Web 服务器插件，用于为客户端与 MobiLink 服务器之间的请求和响应选择发送路径。此插件还实现了负荷平衡和故障转移机制。

抽取

SQL Remote 复制中从统一数据库卸载相应结构和数据的行为。此信息用于初始化远程数据库。

另请参见： “复制”一节第 171 页

触发器

一种特殊形式的存储过程，用户运行修改数据的查询时会自动执行该存储过程。

另请参见：

- “行级触发器”一节第 173 页
- “语句级触发器”一节第 191 页
- “完整性”一节第 187 页

传输规则

QAnywhere 中用于确定何时进行消息传输、传输哪些消息以及应在何时删除消息的逻辑。

窗口

作为分析功能执行对象的行组。一个窗口可以包含一行、多行或所有行的数据，这些数据已根据窗口定义中提供的分组规格进行了分区。窗口会进行移动，以包括为输入中的当前行执行计算所需的行数或行范围。窗口结构的主要优点是，不需要执行附加查询就可以有机会对结果进行分组和分析。

创建者 ID

UltraLite Palm OS 应用程序中一种在创建应用程序时指派的 ID。

存储过程

存储过程是数据库中存储的一组 SQL 指令，用于在数据库服务器上执行一组操作或查询。

代理表

一种本地表，它所包含的元数据可以像访问本地表一样访问远程数据库服务器上的表。

另请参见： “元数据”一节第 192 页

代理 ID

另请参见： “客户端消息存储库 ID”一节第 177 页

代码页

代码页是一种将字符集的字符映射到数字表示的编码，数字表示通常是 0 到 255 之间的一个整数。例如，Windows 代码页 1252 就是一个代码页。就本文档而言，代码页和编码这两个术语可以互换。

另请参见：

- “字符集”一节第 194 页
- “编码”一节第 167 页
- “归类”一节第 173 页

DBA 权限

使用户能够在数据库中执行管理活动的权限级别。DBA 用户在缺省情况下具有 DBA 权限。

另请参见： “数据库管理员 (DBA)”一节第 183 页

dbspace

用于创建更多数据存储空间的附加数据库文件。一个数据库可以包含在最多 13 个独立的文件（一个初始文件和 12 个 dbspace）中。每个表及其索引必须包含在单个数据库文件中。SQL 命令 CREATE DBSPACE 可将新文件添加到数据库中。

另请参见： “数据库文件”一节第 184 页

动态 SQL

执行前由程序以编程方式生成的 SQL。UltraLite 动态 SQL 是一种专用于小型设备的 SQL 变体。

对象树

Sybase Central 中数据库对象的层次。对象树的顶层显示您的 Sybase Central 版本所支持的全部产品。每种产品展开后会显示其自己的对象子树。

另请参见： “Sybase Central”一节第 185 页

EBF

快速错误修正软件。快速错误修正软件是含有一个或多个错误修正软件的软件子集。错误修正软件列在更新程序的发行说明中。错误修正软件更新可能只适用于具有相同版本号的已安装软件。已对该软件执行了一些测试，但该软件尚未进行完全测试。除非您自己已验证了软件的适用性，否则不要随应用程序分发这些文件。

发布

MobiLink 或 SQL Remote 中一种用于标识将要同步的数据的数据库对象。在 MobiLink 中，发布仅存在于客户端。一个发布包括多个项目。SQL Remote 用户可以通过预订发布来接收发布。MobiLink 用户可以通过创建发布的同步预订来同步发布。

另请参见：

- “复制”一节第 171 页
- “项目”一节第 189 页
- “发布更新”一节第 170 页

发布更新

SQL Remote 复制中对一个数据库中的一个或多个发布所做更改的列表。发布更新将作为复制消息的一部分定期发送到远程数据库。

另请参见：

- “复制”一节第 171 页
- “发布”一节第 170 页

发布者

SQL Remote 复制中数据库内可以与其它复制数据库交换复制消息的单个用户。

另请参见： “复制”一节第 171 页。

FILE

SQL Remote 复制中一种使用共享文件来交换复制消息的消息系统。它对测试以及在没有显式消息传送系统的情况下进行的安装很有用。

另请参见“复制”一节第 171 页。

分析树

查询的代数表示。

服务

在 Windows 操作系统上，服务是在运行应用程序的用户 ID 未登录时的应用程序运行方式。

服务器管理请求

一种 QAnywhere 消息，其格式设置为 XML 并发送到 QAnywhere 系统队列，作为一种管理服务器消息存储库或监控 QAnywhere 应用程序的方法。

服务器启动的同步

一种从 MobiLink 服务器启动 MobiLink 同步的方式。

服务器消息存储库

QAnywhere 中在消息传输到客户端消息存储库或 JMS 系统之前服务器上用于临时存储消息的关系数据库。消息通过服务器消息存储库在各客户端之间进行交换。

复制

在物理上不相同的数据库之间共享数据。Sybase 有三种复制技术：MobiLink、SQL Remote 和复制服务器。

复制代理

请参见： “LTM”一节第 178 页

复制服务器

Sybase 的一种基于连接的复制技术，用于与 SQL Anywhere 和 Adaptive Server Enterprise 一起使用。它专用于在一些数据库之间进行接近实时的复制。

另请参见： “LTM”一节第 178 页

复制频率

SQL Remote 复制中一项针对每个远程用户的设置，它决定发布者的消息代理向该远程用户发送复制消息的频率应为多少。

另请参见： “复制”一节第 171 页。

复制消息

SQL Remote 或复制服务器中一种在发布数据库与预订数据库之间发送的通信。消息包含复制系统所需的数据、直通语句及信息。

另请参见：

- “复制”一节第 171 页
- “发布更新”一节第 170 页

隔离级别

一个事务中的操作对其它并发事务中的操作的可见程度。隔离级别有四级，编号依次为 0 至 3。第 3 级提供最高级别的隔离。级别 0 为缺省设置。SQL Anywhere 还支持以下三个快照隔离级别：快照、语句快照和只读语句快照。

另请参见： “快照隔离”一节第 177 页

个人服务器

与客户端应用程序在同一台计算机上运行的数据库服务器。个人数据库服务器通常由单个用户在一台计算机上使用，但它可以支持来自该用户的几个并发连接。

工作表

一种内部存储区域，用于在查询优化过程中存储中间结果。

故障切换

在活动服务器、系统或网络出现故障或意外终止时切换到冗余或备用的服务器、系统或网络。故障转移会自动进行。

关系数据库管理系统 (RDBMS)

一种以相关表的形式存储数据的数据库管理系统。

另请参见： “数据库管理系统 (DBMS)”一节第 183 页

规范化

对数据库模式的改进，目的在于按照基于关系数据库理论的规则消除冗余并改善组织。

归类

定义数据库中文本属性的字符集与排序顺序的组合。对于 SQL Anywhere 数据库，缺省归类取决于运行服务器时所使用的操作系统和语言；例如，英语 Windows 系统上的缺省归类为 1252LATIN1。归类（也称作归类序列）用于对字符串进行比较和排序。

另请参见：

- “字符集”一节第 194 页
- “代码页”一节第 169 页
- “编码”一节第 167 页

行级触发器

每更改一行即执行一次的触发器。

另请参见：

- “触发器”一节第 169 页
- “语句级触发器”一节第 191 页

回退日志

对在每个未提交的事务执行过程中所做更改的记录。当收到 ROLLBACK 请求或者系统出现故障时，未提交的事务会从数据库中回退，将数据库返回其原先的状态。每个事务都有一个单独的回退日志，事务完成时日志会被删除。

另请参见： “事务”一节第 181 页

iAnywhere JDBC 驱动程序

iAnywhere JDBC 驱动程序提供了一个 JDBC 驱动程序，与纯 Java jConnect JDBC 驱动程序相比，该驱动程序拥有一些性能优势和功能优点，但它不是纯 Java 解决方案。建议在大多数情况下使用 iAnywhere JDBC 驱动程序。

另请参见：

- “JDBC”一节第 174 页
- “jConnect”一节第 174 页

InfoMaker

一种报告和数据维护工具，它用于创建复杂的表格、报告、图形、交叉表和表，并创建将这些报告用作构件块的应用程序。

Interactive SQL

一种 SQL Anywhere 应用程序，用于查询和更改数据库中的数据以及修改数据库的结构。Interactive SQL 不但提供了一个用于输入 SQL 语句的窗格，还提供了一些用于返回有关查询处理过程的信息和结果集的窗格。

JAR 文件

Java 档案文件。一种压缩的文件格式，由一个或多个用于 Java 应用程序的包的集合组成。它将安装和运行 Java 程序所需的全部资源都放在一个压缩文件中。

Java 类

Java 中的主要代码结构单元。它是组合在一起的过程和变量的集合，将过程和变量组合在一起的原因是它们都与某个特定的可识别类别有关。

jConnect

JavaSoft JDBC 标准的 Java 实现。它为 Java 开发人员提供多层和异类环境中的本地数据库访问。但在大多数情况下，iAnywhere JDBC 驱动程序是首选的 JDBC 驱动程序。

另请参见：

- “JDBC”一节第 174 页
- “iAnywhere JDBC 驱动程序”一节第 173 页

JDBC

Java 数据库连接。一种 SQL 语言编程接口，它允许 Java 应用程序访问关系数据。首选的 JDBC 驱动程序是 iAnywhere JDBC 驱动程序。

另请参见：

- “jConnect”一节第 174 页
- “iAnywhere JDBC 驱动程序”一节第 173 页

基表

永久性的数据表。有时为区别于临时表和视图，会将这种表称作**基表**。

另请参见：

- “临时表”一节第 177 页
- “视图”一节第 181 页

基于会话的同步

一种同步类型，这种同步会使数据表示在统一数据库和远程数据库都一致。MobiLink 基于会话。

基于脚本的上载

MobiLink 中一种将上载过程自定义为使用日志文件的替代方法的方式。

基于 SQL 的同步

MobiLink 中一种使用 MobiLink 事件将表数据与支持 MobiLink 的统一数据库进行同步的方式。对于基于 SQL 的同步，可以直接使用 SQL，也可以使用面向 Java 和 .NET 平台的 MobiLink 服务器 API 返回 SQL。

基于文件的下载

在 MobiLink 中同步数据的一种方式，其中下载以文件的方式进行分发，从而支持脱机分发同步更改。

集成登录

一种登录功能，它允许将同一个用户 ID 和口令用于操作系统登录、网络登录和数据库连接。

监听器

一个程序 (dblsn)，用于 MobiLink 服务器启动的同步。监听器安装在远程设备上，它们被配置为在接收到来自通告程序的信息时启动针对设备的操作。

另请参见： “服务器启动的同步”一节第 171 页

检查点

将对数据库的所有更改都保存到数据库文件中的时间点。在其它时间，所提交的更改仅保存到事务日志中。

检查约束

对列或列集强制实施指定条件的一种限制。

另请参见：

- “约束”一节第 193 页
- “外键约束”一节第 187 页
- “主键约束”一节第 194 页
- “唯一约束”一节第 188 页

脚本

MobiLink 中为处理 MobiLink 事件而编写的代码。脚本通过编程方式控制数据交换，以满足业务需要。

另请参见： “事件模型”一节第 181 页

脚本版本

MobiLink 中为创建同步而一起应用的一组同步脚本。

校验

测试数据库、表或索引是否受到特定类型的文件损坏。

校验和

随数据库页本身一起记录的计算出的数据库页位数。校验和能够确保数据库页写入磁盘时位数相符，因此数据库管理系统可以通过它来验证数据库页的完整性。如果计数相符，即认为数据库页已成功写入。

镜像日志

另请参见： “事务日志镜像”一节第 182 页

角色

概念性数据库建模中从一个角度描述某种关系的动词或短语。您可以用两个角色来描述每种关系。例如，"包含" 和 "隶属于" 便是角色。

角色名

外键的名称。由于它命名外表和主表之间的关系，因此称作角色名。缺省情况下，角色名就是表名，除非其它外键已经使用该名称（在这种情况下，缺省的角色名是表名后接一个三位的唯一数字）。也可以自己创建角色名。

另请参见： “外键”一节第 186 页

局部临时表

一种临时表，仅在复合语句执行期间或连接结束之前存在。当您只需要将数据集装载一次时，局部临时表非常有用。缺省情况下，行会在提交时被删除。

另请参见：

- “临时表”一节第 177 页
- “全局临时表”一节第 180 页

客户端/服务器

一种软件体系结构，在这种体系结构中，一个应用程序（客户端）从另一个应用程序（服务器）获取信息并向该应用程序发送信息。这两个应用程序常位于通过网络连接的不同计算机上。

客户端消息存储库

QAnywhere 中一种用于在远程设备上存储消息的 SQL Anywhere 数据库。

客户端消息存储库 ID

QAnywhere 中一种对客户端消息存储库进行唯一标识的 MobiLink 远程 ID。

快照隔离

一种为发出读请求的事务返回数据的已提交版本的隔离级别。SQL Anywhere 提供了以下三种快照隔离级别：快照、语句快照和只读语句快照。使用快照隔离时，读操作不会阻塞写操作。

另请参见： “隔离级别”一节第 172 页

连接

关系系统中的一种基本操作，它通过比较指定列中的值将两个或更多个表中的行链接在一起。

连接 ID

用于标识客户端应用程序与数据库之间给定连接的唯一编号。可以使用以下 SQL 语句来确定当前连接 ID：

```
SELECT CONNECTION_PROPERTY( 'Number' );
```

连接类型

SQL Anywhere 提供了四种类型的连接：交叉连接、键连接、自然连接和使用 ON 子句的连接。

另请参见： “连接”一节第 177 页

连接配置

连接到数据库所需的一组参数，如用户名、口令和服务器名称，它们在存储后即可方便地使用。

连接启动的同步

一种 MobiLink 服务器启动的同步，在这种同步下，连接发生变化时会启动同步。

另请参见： “服务器启动的同步”一节第 171 页

连接条件

一种影响连接结果的限制。您可以通过紧跟在连接语句的后面插入 ON 子句或 WHERE 子句来指定连接条件。对于自然连接和关键连接，SQL Anywhere 会生成连接条件。

另请参见：

- “连接”一节第 177 页
- “生成的连接条件”一节第 182 页

临时表

为临时存储数据而创建的表。有两种类型：全局临时表和局部临时表。

另请参见：

- “局部临时表”一节第 176 页
- “全局临时表”一节第 180 页

LTM

日志传送管理器（Log Transfer Manager，简称 LTM）也称作复制代理。LTM 是一个与 Replication Server 一起使用的程序，它读取数据库事务日志并将提交的更改发送到 Sybase 复制服务器。

请参见： “复制服务器”一节第 172 页

轮询

在 MobiLink 服务器启动的同步中，轻量级轮询器（例如 MobiLink 监听器）从通告程序请求推式通知的方式。

另请参见： “服务器启动的同步”一节第 171 页

逻辑索引

指向物理索引的引用（指针）。磁盘上不存储逻辑索引的索引结构。

命令文件

包含 SQL 语句的文本文件。命令文件可以手工建立，也可以通过数据库实用程序自动建立。例如，dbunload 实用程序会创建一个命令文件，其中包含重新创建给定数据库所需的 SQL 语句。

MobiLink

一种基于会话的同步技术，其设计用途是将 UltraLite 和 SQL Anywhere 远程数据库与统一数据库同步。

另请参见：

- “统一数据库”一节第 185 页
- “同步”一节第 185 页
- “UltraLite”一节第 186 页

MobiLink 服务器

运行 MobiLink 同步的计算机程序，即 mlsrv11。

MobiLink 监控器

一种用于监控 MobiLink 同步的图形化工具。

MobiLink 客户端

有两种 MobiLink 客户端。对于 SQL Anywhere 远程数据库，MobiLink 客户端是 dbmlsync 命令行实用程序。对于 UltraLite 远程数据库，MobiLink 客户端内置于 UltraLite 运行时库中。

MobiLink 系统表

MobiLink 同步所需的系统表。它们由 MobiLink 安装程序脚本安装到 MobiLink 统一数据库中。

MobiLink 用户

MobiLink 用户用于与 MobiLink 服务器进行连接。在远程数据库上创建 MobiLink 用户，然后在统一数据库中注册该用户。MobiLink 用户名完全独立于数据库用户名。

模式

数据库的结构，其中包括表、列和索引以及它们之间的关系。

内连接

一种连接，在这种连接中，仅当两个表都满足连接条件时行才会出现在结果集中。内连接是缺省设置。

另请参见：

- “连接”一节第 177 页
- “外连接”一节第 187 页

ODBC

开放式数据库连接。一种用于与数据库管理系统连接的标准 Windows 接口。ODBC 是 SQL Anywhere 所支持的几种接口之一。

ODBC 管理器

一种随 Windows 操作系统提供的 Microsoft 程序，用于设置 ODBC 数据源。

ODBC 数据源

用户要通过 ODBC 访问的数据的规范以及获取该数据时所需的信息。

PDB

Palm 数据库文件。

PowerDesigner

一种数据库建模应用程序。PowerDesigner 为设计数据库或数据仓库提供了结构化的方法。SQL Anywhere 包括 PowerDesigner 的 Physical Data Model 组件。

PowerJ

一种 Sybase 产品，用于开发 Java 应用程序。

QAnywhere

应用程序到应用程序的消息传递（包括移动设备到移动设备和移动设备与企业之间的消息传递），它使在移动或无线设备上运行的自定义程序能够与处在中央位置的服务器应用程序进行通信。

QAnywhere 代理

QAnywhere 中一种运行在客户端设备上的进程，用于监控客户端消息存储库和确定应在何时传输消息。

嵌入式 SQL

一种 C 语言程序编程接口。SQL Anywhere 嵌入式 SQL 是 ANSI 和 IBM 标准的实现。

轻量级轮询器

在 MobiLink 服务器启动的同步中，轮询来自 MobiLink 服务器的推式通知的设备应用程序。

另请参见： “服务器启动的同步”一节第 171 页

全局临时表

一种临时表，在被显式地删除之前，其数据定义对所有用户都可见。全局临时表允许用户各自打开一个表的相同实例。缺省情况下，行在提交时被删除，并且始终是在连接结束时被删除。

另请参见：

- “临时表”一节第 177 页
- “局部临时表”一节第 176 页

日志文件

SQL Anywhere 所维护的事务日志。该日志文件用于确保在出现系统或介质故障时可以恢复数据库、提高数据库性能以及使用 SQL Remote 实现数据复制。

另请参见：

- “事务日志”一节第 181 页
- “事务日志镜像”一节第 182 页
- “完全备份”一节第 187 页

散列

散列是一种将索引条目转化为键的索引优化。索引散列旨在通过将足够的行实际数据与其行 ID 包括在一起，以避免进行先查找行、后装载行然后再将行解出才能得出索引值的高开销操作。

上载

同步过程的一个阶段，在此阶段数据从远程数据库传送到统一数据库。

设备跟踪

在 MobiLink 服务器启动的同步中，允许使用标识设备的 MobiLink 用户名来对消息进行寻址的功能。

另请参见： “服务器启动的同步”一节第 171 页

实例化视图

实例化视图是指已计算并已存储在磁盘上的视图。实例化视图同时具有视图的特征（使用查询说明进行定义）和表的特征（可以对其执行大多数表操作）。

另请参见：

- “基表”一节第 174 页
- “视图”一节第 181 页

世代号

MobiLink 中的一种机制，用于强制远程数据库先上载数据，然后再应用任何其它下载文件。

另请参见： “基于文件的下载”一节第 175 页

事件模型

MobiLink 中组成同步的事件（如 begin_synchronization 和 download_cursor）序列。如果为事件创建了脚本，则会调用事件。

视图

一种作为对象存储在数据库中的 SELECT 语句。它使用户能够看到一个或多个表中的行子集或列子集。每当用户使用特定表或表组合的视图时，都将利用存储在这些表中的信息重新计算视图。视图对确保安全以及定制数据库信息的外观来使数据访问简单明了有帮助。

事务

组成一个逻辑工作单元的 SQL 语句序列。事务要么全部得到处理，要么根本不做处理。SQL Anywhere 支持事务处理，并内置了锁定功能，使并发事务能够访问数据库而又不损坏数据。事务要么以 COMMIT 语句结束，该语句使对数据的更改成为永久性更改；要么以 ROLLBACK 语句结束，该语句撤消在事务执行过程中所做的全部更改。

事务日志

一种按进行更改的顺序存储对数据库所做全部更改的文件。它会提高性能并支持在数据库文件损坏时恢复数据。

事务日志镜像

同时维护的事务日志文件的完全相同副本（可选）。每当数据库更改写入事务日志文件时，也会同时写入事务日志镜像文件。

镜像文件应该与事务日志保留在不同的设备上，这样在任意设备出现故障时，日志的其它副本会确保数据可以安全地恢复。

另请参见： “事务日志”一节第 181 页

事务完整性

MobiLink 中对整个同步系统事务的有保证维护。要么同步整个事务，要么不对事务的任何部分进行同步。

生成的连接条件

一种自动生成的对连接结果的限制。有两种类型：关键和自然。指定 KEY JOIN 或指定关键字 JOIN 但不使用关键字 CROSS、NATURAL 或 ON 时，会生成关键连接。对于关键连接，所生成的连接条件取决于表之间的外键关系。指定 NATURAL JOIN 时会生成自然连接；所生成的连接条件基于两个表中的公用列名。

另请参见：

- “连接”一节第 177 页
- “连接条件”一节第 177 页

受保护的功能

数据库服务器启动时由 -sf 选项指定的功能，该数据库服务器上运行的任何数据库都无法使用该功能。

授权选项

一种权限级别，它允许用户向其他用户授予权限。

数据操作语言 (DML)

用于操作数据库中数据的 SQL 语句子集。DML 语句可以检索、插入、更新和删除数据库中的数据。

数据定义语言 (DDL)

用于定义数据库中数据结构的 SQL 语句子集。DDL 语句可以创建、修改和删除数据库对象（如表和用户）。

数据类型

数据的格式，如 CHAR 或 NUMERIC。在 ANSI SQL 标准中，数据类型也可以包括对大小、字符集和归类的限制。

另请参见： “域”一节第 191 页

数据立方体

一种多维结果集，每一维都以不同的方式对相同的结果进行分组和排序。数据立方体提供了有关数据的综合性信息，如果不使用数据立方体，要获得同样的信息就必须进行自连接查询和相关子查询。数据立方体是 OLAP 功能的一部分。

数据库

通过主键和外键关联的表的集合。表包含数据库中的信息。表和键一起定义数据库的结构。数据库管理系统会访问此信息。

另请参见：

- “外键”一节第 186 页
- “主键”一节第 194 页
- “数据库管理系统 (DBMS)”一节第 183 页
- “关系数据库管理系统 (RDBMS)”一节第 172 页

数据库对象

包含或接收信息的数据库组件。表、索引、视图、过程和触发器便是数据库对象。

数据库服务器

对所有针对数据库信息的访问进行管理的计算机程序。SQL Anywhere 提供了两种类型的服务器：网络服务器和个人服务器。

数据库管理系统 (DBMS)

用于创建和使用数据库的程序的集合。

另请参见： “关系数据库管理系统 (RDBMS)”一节第 172 页

数据库管理员 (DBA)

具有维护数据库所需权限的用户。DBA 通常负责对数据库模式的所有更改以及管理用户和组。数据库管理员角色自动内置于数据库中，其用户 ID 为 DBA，口令是 sql。

数据库连接

客户端应用程序与数据库之间的通信渠道。必须具有有效的用户 ID 和口令才能建立连接。为用户 ID 授予的特权决定了在连接过程中可以执行的操作。

数据库名称

服务器装载数据库时为数据库指定的名称。缺省数据库名是初始数据库文件的文件名（不含扩展名）。

另请参见： “数据库文件”一节第 184 页

数据库所有者 (dbo)

一种特殊的用户，他拥有不归 SYS 所有的系统对象。

另请参见：

- “数据库管理员 (DBA)”一节第 183 页
- “SYS”一节第 185 页

数据库文件

数据库保存在一个或多个数据库文件中。其中一个是初始文件，后面的文件称作 dbspace。每个表（包括其索引）都必须包含在单个数据库文件中。

另请参见： “dbspace”一节第 170 页

死锁

一组事务会进入的一种特殊状态，在该状态下这些事务都不能继续执行。

SQL

用于与关系数据库进行通信的语言。ANSI 定义了 SQL 的标准，其最新标准是 SQL-2003。SQL 的非官方全称是结构化查询语言。

SQL Anywhere

SQL Anywhere 的关系数据库服务器组件，专供在移动和嵌入式环境中使用，或作为中小型企业的服务器使用。SQL Anywhere 也是包含 SQL Anywhere RDBMS、UltraLite RDBMS、MobiLink 同步软件和其它组件的软件包的名称。

SQL Remote

一种基于消息的数据复制技术，用于在统一数据库与远程数据库之间进行双向复制。统一数据库和远程数据库必须是 SQL Anywhere。

SQL 语句

包含用于将指令传递给 DBMS 的 SQL 关键字的字符串。

另请参见：

- “模式”一节第 179 页
- “SQL”一节第 184 页
- “数据库管理系统 (DBMS)”一节第 183 页

锁定

一种在同时执行多个事务的过程中保护数据完整性的并发控制机制。SQL Anywhere 会自动应用锁以防止两个连接同时更改同一数据，并防止其它连接读取正接受更改的数据。

您可以通过设置隔离级别来控制锁定。

另请参见：

- “隔离级别”一节第 172 页
- “并发”一节第 167 页
- “完整性”一节第 187 页

索引

一组已排序的、与基表中的一个或多个列关联的键和指针。在表中一个或多个列上设置索引可以提高性能。

Sybase Central

一种数据库管理工具，通过图形用户界面提供 SQL Anywhere 数据库设置、属性和实用程序。Sybase Central 也可用于管理其它 Sybase 产品，其中包括 MobiLink。

SYS

一种拥有大多数系统对象的特殊用户。无法以 SYS 身份登录。

统一数据库

在分布式数据库环境中，是指用于存储数据主副本的数据库。出现冲突或差异时，将把统一数据库视为具有数据的主副本。

另请参见：

- “同步”一节第 185 页
- “复制”一节第 171 页

通信流

MobiLink 中 MobiLink 客户端与 MobiLink 服务器之间进行通信时所使用的网络协议。

通告程序

一种由 MobiLink 服务器启动的同步使用的程序。通告程序集成在 MobiLink 服务器中。它们会检查统一数据库是否有推式请求，并发送推式通知。

另请参见：

- “服务器启动的同步”一节第 171 页
- “监听器”一节第 175 页

同步

利用 MobiLink 技术在数据库之间复制数据的过程。

在 SQL Remote 中，同步专指以初始数据集初始化远程数据库的过程。

另请参见：

- “MobiLink”一节第 178 页
- “SQL Remote”一节第 184 页

推式请求

在 MobiLink 服务器启动的同步中，通告程序通过检查它来确定推式通知是否需要发送到设备的结果集中的一行值。

另请参见： “服务器启动的同步”一节第 171 页

推式通知

QAnywhere 中一种从服务器传送到 QAnywhere 客户端的特殊消息，用于提示客户端启动消息传输。在 MobiLink 服务器启动的同步中，从通告程序传送到包含推式请求数据和内部信息的设备的特殊消息。

另请参见：

- “QAnywhere”一节第 180 页
- “服务器启动的同步”一节第 171 页

UltraLite

一种针对小型设备、移动设备和嵌入式设备进行了优化的数据库。所面向的平台包括手机、传呼机和个人记事本。

UltraLite 运行时

一种过程中关系数据库管理系统，其中包括一个内置 MobiLink 同步客户端。每个 UltraLite 编程接口使用的库以及 UltraLite 引擎中都包括 UltraLite 运行时。

外表

包含外键的表。

另请参见： “外键”一节第 186 页

外部登录

与远程服务器通信时使用的替代登录名和口令。缺省情况下，SQL Anywhere 每次代表其客户端连接到远程服务器时都会使用这些客户端的名称和口令。但是，您可以通过创建外部登录来替换这一缺省设置。外部登录是指与远程服务器通信时使用的替代登录名和口令。

外键

一个表中复制另一个表中主键值的一个或多个列。外键建立表间的关系。

另请参见：

- “主键”一节第 194 页
- “外表”一节第 186 页

外键约束

对单个列或一组列的限制，指定表中的数据与某个其它表中数据的关系。对列集施加外键约束可使这些列成为外键。

另请参见：

- “约束”一节第 193 页
- “检查约束”一节第 175 页
- “主键约束”一节第 194 页
- “唯一约束”一节第 188 页

外连接

一种保留表中所有行的连接。SQL Anywhere 支持左、右和完全外连接。左外连接保留表中位于连接运算符左侧的行，当右表中的行不满足连接条件时，它将返回空值。完全外连接保留两个表中的所有行。

另请参见：

- “连接”一节第 177 页
- “内连接”一节第 179 页

完全备份

对整个数据库和事务日志（可选）的备份。完全备份包含数据库中的所有信息，因此可以在系统或介质出现故障时提供保护。

另请参见： “增量备份”一节第 193 页

完整性

遵守完整性规则的情况，完整性规则确保数据正确并准确，而且数据库的关系结构保持不变。

另请参见： “参照完整性”一节第 168 页

网关

一种 MobiLink 对象，存储在 MobiLink 系统表或通告程序属性文件中，包含有关如何发送用于服务器启动同步的消息的信息。

另请参见： “服务器启动的同步”一节第 171 页

网络服务器

从共享公共网络的计算机接受连接的数据库服务器。

另请参见： “个人服务器”一节第 172 页

网络协议

通信类型，如 TCP/IP 或 HTTP。

维护版本

维护版本是一套完整的软件，它升级已安装的具有相同主版本号的较早版本的软件（版本号格式是 *major.minor.patch.build*）。升级程序的发行说明中列出了错误修正软件和其它更改。

唯一约束

对某个列或一组列的限制，它要求所有非空值都各不相同。一个表可以有多个唯一约束。

另请参见：

- “外键约束”一节第 187 页
- “主键约束”一节第 194 页
- “约束”一节第 193 页

谓语

一种条件表达式，可以选择性地将其与逻辑运算符 AND 和 OR 组合在一起，以组成 WHERE 或 HAVING 子句中的条件集。在 SQL 中，求值结果为 UNKNOWN 的谓语将解释为 FALSE。

位数组

位数组是一种用于有效率地存储位序列的数组数据结构。位数组与字符串类似，不同的是其各个部分由 0（零）和 1（一）而不是字符组成。位数组通常用于保存一串布尔值。

Windows

Microsoft Windows 操作系统系列，如 Windows Vista、Windows XP 和 Windows 200x。

Windows CE

请参见“Windows Mobile”一节第 188 页。

Windows Mobile

Microsoft 为移动设备制造的操作系统系列。

文件定义数据库

MobiLink 中一种用于创建下载文件的 SQL Anywhere 数据库。

另请参见： “基于文件的下载”一节第 175 页

物理索引

索引存储在磁盘上的实际索引结构。

系统表

一种表，由 SYS 或 dbo 拥有，用于保存元数据。系统表也称作数据字典表，由数据库服务器创建并维护。

系统对象

由 SYS 或 dbo 拥有的数据库对象。

系统视图

存在于每一个数据库中的一种视图，它以易于理解的格式表示系统表中包含的信息。

下载

同步过程的一个阶段，在此阶段数据从统一数据库传送到远程数据库。

相关名

查询的 FROM 子句中使用的表或视图的名称—要么是表或视图的原始名称，要么是在 FROM 子句中定义的替代名称。

项目

在 MobiLink 或 SQL Remote 中，项目是表示整个表或表中行和列子集的数据库对象。项目在发布中组合在一起。

另请参见：

- “复制”一节第 171 页
- “发布”一节第 170 页

消息存储库

QAnywhere 中客户端和服务器设备上存储消息的数据库。

另请参见：

- “客户端消息存储库”一节第 176 页
- “服务器消息存储库”一节第 171 页

消息类型

SQL Remote 复制中指定远程用户与统一数据库发布者通信方式的数据库对象。一个统一数据库可能定义了几种消息类型，这样一来，不同的远程用户就可以使用不同的消息系统与统一数据库进行通信。

另请参见：

- “复制”一节第 171 页
- “统一数据库”一节第 185 页

消息日志

可存储来自数据库服务器或 MobiLink 服务器等应用程序的消息的日志。此类信息还可以出现在消息窗口中或记录到文件中。消息日志包括信息性消息、错误、警告以及来自 MESSAGE 语句的消息。

消息系统

SQL Remote 复制中用于在统一数据库与远程数据库之间交换消息的协议。SQL Anywhere 包括对以下消息系统的支持：FILE、FTP 和 SMTP。

另请参见：

- “复制”一节第 171 页
- “FILE”一节第 171 页

卸载

卸载数据库时会将数据库的结构和/或数据导出到文本文件（如果是结构，则导出到 SQL 命令文件中；如果是数据，则导出到 ASCII 逗号分隔文件中）。使用卸载实用程序来卸载数据库。

此外，您也可以使用 UNLOAD 语句卸载数据的选定部分。

性能统计

反映数据库系统性能的值。例如，CURRREAD 统计表示数据库服务器已发出但尚未完成的文件读取次数。

业务规则

基于实际要求的准则。通常，业务规则通过检查约束、用户定义数据类型以及事务的正确使用来实现。

另请参见：

- “约束”一节第 193 页
- “用户定义数据类型”一节第 191 页

引用对象

一种对象（如视图），其定义直接引用数据库中的另一个对象（如表）。

另请参见： “外键”一节第 186 页

用户定义数据类型

请参见“域”一节第 191 页。

游标

指向结果集的已命名链接，用于通过编程接口访问和更新行。在 SQL Anywhere 中，游标支持在查询结果中进行向前和向后移动。游标由两部分组成：游标结果集（通常由 SELECT 语句定义）和游标位置。

另请参见：

- “游标结果集”一节第 191 页
- “游标位置”一节第 191 页

游标结果集

与游标关联的查询所得到的行集。

另请参见：

- “游标”一节第 191 页
- “游标位置”一节第 191 页

游标位置

指向游标结果集中一个行的指针。

另请参见：

- “游标”一节第 191 页
- “游标结果集”一节第 191 页

语句级触发器

在整个触发语句完成后执行的触发器。

另请参见：

- “触发器”一节第 169 页
- “行级触发器”一节第 173 页

域

内置数据类型的别名，其中包括适用的精度值和小数位值，还可以选择是否包括 DEFAULT 值和 CHECK 条件。SQL Anywhere 中预定义了一些域，如货币数据类型。也称作用户定义数据类型。

另请参见： “数据类型”一节第 182 页

预订

MobiLink 同步中发布与 MobiLink 用户之间的客户端数据库中的一个链接，它使发布所描述的数据能够得到同步。

SQL Remote 复制中发布与远程用户之间的一种链接，它使用户能够与统一数据库交换该发布上的更新。

另请参见：

- “发布”一节第 170 页
- “MobiLink 用户”一节第 179 页

元数据

数据的数据。元数据描述其它数据的性质和内容。

另请参见： “模式”一节第 179 页

原子事务

保证成功完成或保证根本不予完成的事务。如果错误使原子事务的一部分无法完成，则将回退事务以防止数据库处于不一致的状态。

REMOTE DBA 特权

在 SQL Remote 中，消息代理 (dbremote) 所需的权限级别。MobiLink 中 SQL Anywhere 同步客户端 (dbmlsync) 所需的权限级别。当消息代理或同步客户端作为具有该权限的用户建立连接时，它将具有完全的 DBA 访问权。如果不是通过消息代理或同步客户端进行连接，则该用户 ID 将不具有附加权限。

另请参见： “DBA 权限”一节第 170 页

远程 ID

SQL Anywhere 和 UltraLite 数据库中一种由 MobiLink 使用的唯一标识符。远程 ID 初始情况下设置为 NULL，在数据库第一次同步期间将设置为 GUID。

远程数据库

MobiLink 或 SQL Remote 中一种与统一数据库交换数据的数据库。远程数据库可以共享统一数据库中的全部或部分数据。

另请参见：

- “同步”一节第 185 页
- “统一数据库”一节第 185 页

约束

对特定数据库对象（如表或列）中所包含值的限制。例如，列可以具有唯一性约束，该约束要求该列中的所有值互不相同。表可以具有外键约束，该约束指定该表中的信息与某个其它表中数据的关系。

另请参见：

- “检查约束”一节第 175 页
- “外键约束”一节第 187 页
- “主键约束”一节第 194 页
- “唯一约束”一节第 188 页

运营公司

一种 MobiLink 对象，存储在 MobiLink 系统表或通告程序属性文件中，包含有关供服务器启动的同步使用的公共运营公司的信息。

另请参见： “服务器启动的同步”一节第 171 页

增量备份

仅包含事务日志的备份，通常在两次完全备份之间使用。

另请参见： “事务日志”一节第 181 页

争用

为获取资源而竞争的行为。例如，就数据库而言，如果有两个或更多个用户试图编辑数据库的同一行，就会为获得编辑该行的权利而发生争用。

正则表达式

正则表达式是字符、通配符和运算符的序列，用于定义某种模式以在字符串内进行搜索。

直方图

直方图是列统计信息最重要的组成部分，是一种表示数据分布的方式。SQL Anywhere 维护直方图以为优化程序提供有关列值分布情况的统计信息。

直接行处理

MobiLink 中一种用于将表数据同步到 MobiLink 支持的统一数据库以外的数据源的方法。使用直接行处理时，上载和下载都可以实现。

另请参见：

- “统一数据库”一节第 185 页
- “基于 SQL 的同步”一节第 175 页

主表

包含外键关系中的主键的表。

主键

其值唯一标识表中各行的一个列或多个列。

另请参见： “外键”一节第 186 页

主键约束

一种对主键列的唯一性约束。一个表只能有一个主键约束。

另请参见：

- “约束”一节第 193 页
- “检查约束”一节第 175 页
- “外键约束”一节第 187 页
- “唯一约束”一节第 188 页
- “完整性”一节第 187 页

子查询

嵌套在 SELECT、INSERT、UPDATE 或 DELETE 语句或者其它子查询中的 SELECT 语句。

有两种类型的子查询：相关子查询和嵌套子查询。

字符串

字符串是以单引号围起的字符序列。

字符集

字符集是一组符号，包括字母、数字、空格和其它符号。字符集的一个例子是 ISO-8859-1，又称作 Latin1。

另请参见：

- “代码页”一节第 169 页
- “编码”一节第 167 页
- “归类”一节第 173 页

索引

其它

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z