



# MobiLink 入门

上次修改时间 2007 年 3 月

## 版权和商标

版权所有 (c) 2007 iAnywhere Solutions, Inc. 部分版权所有 (c) 2007 Sybase, Inc. 保留所有权利。

iAnywhere Solutions, Inc. 是 Sybase, Inc. 的子公司。

iAnywhere 授权您将本文档用于参考、教育以及其它非商业用途，前提是 (1) 所有副本均含有文档中的本篇和所有其它版权和所有权声明；(2) 不要企图将本文档据为己有；(3) 不对本文档进行修改。未经 iAnywhere 明确的事先书面许可，不得出版或发布本文档及其任何部分。

本文档对 iAnywhere 采取或避免采取的任何活动均不做任何承诺，iAnywhere 可自行更改文档内容，恕不另行通知。除非在与 iAnywhere 达成的书面协议中另行规定，否则会原样提供文档，iAnywhere 对文档的使用及其可能包含的不准确性不承担任何责任。

iAnywhere (R)、Sybase (R) 以及在 <http://www.iAnywhere.com/trademarks> 上所列出商标均为 Sybase, Inc. 或其子公司的商标。(R) 表示在美国注册。

Java 以及所有基于 Java 的商标均为 Sun Microsystems, Inc. 在美国和其它国家/地区的商标或注册商标。

文中提及的所有其它公司和产品名可能是与其相关的各个公司的商标。

---

---

# 目录

关于本手册 .....	vii
SQL Anywhere 文档 .....	viii
文档约定 .....	x
查找详细信息并提供反馈 .....	xiv
<b>I. 使用 MobiLink 技术 .....</b>	<b>1</b>
<b>MobiLink 同步简介 .....</b>	<b>3</b>
MobiLink 快速入门 .....	4
同步系统的组成部分 .....	7
中央数据源 .....	9
MobiLink 客户端 .....	10
MobiLink 服务器 .....	11
同步过程 .....	12
用于编写同步逻辑的选项 .....	18
安全 .....	20
在 Mac OS X 上运行 MobiLink .....	21
<b>MobiLink 模型 .....</b>	<b>23</b>
MobiLink 模型简介 .....	24
创建模型 .....	25
模型模式 .....	28
部署模型 .....	38
<b>II. MobiLink 教程 .....</b>	<b>43</b>
<b>探讨 MobiLink 的 CustDB 示例 .....</b>	<b>45</b>
MobiLink CustDB 教程简介 .....	46
CustDB 安装 .....	48
CustDB 数据库中的表 .....	54
CustDB 示例中的用户 .....	57
同步 CustDB .....	58
维护客户和订单的主键池 .....	62
清除 .....	64

进一步阅读 .....	65
<b>研究 MobiLink Contact 示例 .....</b>	<b>67</b>
Contact 示例教程简介 .....	68
Contact 示例安装 .....	69
Contact 数据库中的表 .....	71
Contact 示例中的用户 .....	73
同步 Contact 示例 .....	74
在 Contact 示例中监控统计信息和错误 .....	80
<b>教程：将 MobiLink 用于 Oracle 10g 统一数据库 .....</b>	<b>81</b>
MobiLink Oracle 教程简介 .....	82
第 1 课：创建数据库 .....	83
第 2 课：启动 MobiLink 服务器 .....	88
第 3 课：启动 MobiLink 同步客户端 .....	89
进一步阅读 .....	90
<b>教程：使用 Java 同步逻辑 .....</b>	<b>91</b>
Java 同步教程简介 .....	92
第 1 课：编译 CustdbScripts Java 类 .....	93
第 2 课：指定类方法来处理事件 .....	95
第 3 课：用 -sl java 运行 MobiLink 服务器 .....	98
第 4 课：测试同步 .....	99
清除 .....	100
进一步阅读 .....	101
<b>教程：使用 .NET 同步逻辑 .....</b>	<b>103</b>
.NET 同步教程简介 .....	104
第 1 课：用 MobiLink 参考编译 CustdbScripts.dll 程序集 .....	105
第 2 课：为事件指定类方法 .....	109
第 3 课：带 -sl dnet 运行 MobiLink .....	112
第 4 课：测试同步 .....	113
清除 .....	115
进一步阅读 .....	116
<b>教程：使用 .NET 和 Java 进行自定义验证 .....</b>	<b>117</b>
MobiLink 自定义验证简介 .....	118
第 1 课：创建用于自定义验证的 Java 或 .NET 类（服务器端） .....	119
第 2 课：为 authenticate_user 事件注册 Java 或 .NET 脚本 .....	122
第 3 课：启动面向 Java 或 .NET 的 MobiLink 服务器 .....	123

---

第 4 课：测试验证 .....	124
清除 .....	125
进一步阅读 .....	126
<b>教程：使用直接行处理 .....</b>	<b>127</b>
直接行处理教程简介 .....	128
第 1 课：建立 MobiLink 统一数据库 .....	129
第 2 课：添加同步脚本 .....	132
第 3 课：为处理直接行处理编写 Java 或 .NET 逻辑 .....	135
第 4 课：启动 MobiLink 服务器 .....	146
第 5 课：建立 MobiLink 客户端 .....	147
第 6 课：同步 .....	149
清除 .....	151
进一步阅读 .....	152
索引 .....	153

---

---

# 关于本手册

## 主题

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。

## 读者

本手册适用于那些想要在其信息系统中添加同步的 SQL Anywhere 及其它关系数据库系统的用户。

## 开始之前

有关 MobiLink 与其它同步和复制技术的比较，请参见“[数据交换技术概述](#)” 《SQL Anywhere 10 - 简介》。

## SQL Anywhere 文档

本手册是 SQL Anywhere 文档集的一部分。本节介绍文档集中包含的手册及其使用方法。

### SQL Anywhere 文档

完整的 SQL Anywhere 文档以两种形式提供：一种是联机文档，它合并了所有手册；另一种是 PDF 文件，每本手册都有单独的 PDF 文件。两种形式的文档均包含相同的信息，并且都包括以下手册：

- ◆ **SQL Anywhere 10 - 简介** 本手册介绍 SQL Anywhere 10——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、桌面环境、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 10 - 更改和升级** 本手册介绍 SQL Anywhere 10 以及该软件以前版本中的新功能。
- ◆ **SQL Anywhere 服务器 - 数据库管理** 本手册提供了与运行、管理和配置 SQL Anywhere 数据库相关的资料。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 - 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。
- ◆ **SQL Anywhere 服务器 - SQL 参考** 本手册为 SQL Anywhere 使用的 SQL 语言提供了一套完整的参考资料。它同时介绍了 SQL Anywhere 系统视图和过程。
- ◆ **SQL Anywhere 服务器 - 编程** 本手册介绍如何使用 C、C++、Java 编程语言以及 Visual Studio .NET 建立和部署数据库应用程序。使用 Visual Basic 和 PowerBuilder 等工具的用户可以使用这些工具提供的编程接口。
- ◆ **SQL Anywhere 10 - 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。
- ◆ **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 - 服务器管理** 本手册说明如何设置和管理 MobiLink 应用程序。
- ◆ **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。
- ◆ **MobiLink - 服务器启动的同步** 本手册介绍 MobiLink 服务器启动的同步，利用这项 MobiLink 功能，可从统一数据库启动同步或其它远程操作。
- ◆ **QAnywhere** 本手册将介绍 QAnywhere，该产品是一个消息传递平台，支持移动和无线客户端以及传统桌面和膝上型客户端。
- ◆ **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。

- ◆ **SQL Anywhere 10 - 上下文相关帮助** 本手册提供 [连接] 对话框、查询编辑器、MobiLink 监控器、SQL Anywhere 控制台实用程序、索引顾问和 Interactive SQL 的上下文相关帮助。
- ◆ **UltraLite - 数据库管理和参考** 本手册介绍用于小型设备的 UltraLite 数据库系统。
- ◆ **UltraLite - AppForge 编程** 本手册介绍 UltraLite for AppForge。利用 UltraLite for AppForge，用户可以开发数据库应用程序，并将它们部署到运行 Palm OS、Symbian OS 或 Windows CE 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。
- ◆ **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 CE 或 Windows XP 的手持式设备、移动设备或嵌入式设备。
- ◆ **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。

## 文档格式

SQL Anywhere 按下列形式提供文档：

- ◆ **联机文档** 联机文档包含完整的 SQL Anywhere 文档，其中包括手册和 SQL Anywhere 工具的上下文相关帮助。每次发布产品的维护版本时都会对联机文档进行更新，因此联机文档是最完整和最新的文档来源。

若要在 Windows 操作系统中访问联机文档，请选择 [开始] ► [程序] ► [SQL Anywhere 10] ► [联机手册]。可以使用左窗格中的 HTML 帮助目录、索引和搜索功能，以及右窗格中的链接和菜单来浏览联机文档。

若要在 Unix 操作系统中访问联机文档，请查看 SQL Anywhere 安装目录下或安装 CD 上的 HTML 文档。

- ◆ **PDF 文件** 整套 SQL Anywhere 手册会以一组 Adobe Portable Document Format (pdf) 文件的形式提供，它们可以通过 Adobe Reader 进行查看。

在 Windows 上，可通过每页顶部的 PDF 链接从联机手册查看 PDF 手册，或者从 Windows 的 [开始] 菜单（[开始] ► [程序] ► [SQL Anywhere 10] ► [联机手册 - PDF 格式]）来查看。

在 Unix 上，可在安装 CD 上查看 PDF 手册。

## 文档约定

本节列出了本文档中使用的印刷和图形约定。

### 语约定

在 SQL 语法说明中，使用了以下约定：

- ◆ **关键字** 所有 SQL 关键字都以大写字母显示，就像下例中的 ALTER TABLE：

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

- ◆ **占位符** 对于必须替换为相应的标识符或表达式的项，其显示方式就像下例中的 *owner* 和 *table-name*：

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

- ◆ **重复项** 对于重复项列表，只显示列表中的一个元素，后跟省略号（三个句点），如以下示例中的 *column-constraint*：

```
ADD column-definition [ column-constraint, ... ]
```

允许指定一个或多个列表元素。在此示例中，如果指定了多个元素，则必须用逗号将它们隔开。

- ◆ **可选部分** 语句的可选部分括在方括号内。

```
RELEASE SAVEPOINT [ savepoint-name ]
```

这些方括号表示 *savepoint-name* 是可选项。不应键入方括号。

- ◆ **选项** 如果不必选择项目列表的项目或者只能选中一个，则用垂直条分隔这些项目，并将列表括在方括号内。

```
[ ASC | DESC ]
```

例如，可以从 ASC、DESC 中任选一个，或者一个也不选。不应键入方括号。

- ◆ **二选一选项** 如果必须明确选择一个选项，则会将替换选项括在大括号内，并用一条竖线分隔选项。

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

如果使用了 QUOTES 选项，则必须提供 ON 和 OFF 两者之一。不应键入方括号和大括号。

### 操作系统约定

- ◆ **Windows** 用于台式计算机和膝上型计算机的 Microsoft Windows 操作系统系列。Windows 系列包括 Windows Vista 和 Windows XP。
- ◆ **Windows CE** 由 Microsoft Windows CE 模块操作系统构建的平台，其中包括 Windows Mobile 和 Windows Embedded CE 平台。

Windows Mobile 是在 Windows CE 的基础上构建的。它提供 Windows 用户界面和附加功能，例如小版本的应用程序（如 Word 和 Excel）。Windows Mobile 最常见于移动设备。

SQL Anywhere 中的限制或变动通常基于基础操作系统 (Windows CE)，很少基于使用的特定变型 (Windows Mobile)。

- ◆ **Unix** 除非另外指定，否则 Unix 是指 Linux 和 Unix 平台。

## 文件名约定

当说明与操作系统相关的任务和功能（如路径和文件名）时，本文档通常采用 Windows 约定。大多数情况下，还会简单地转换为用于其它操作系统的语法。

- ◆ **目录和路径名** 文档通常使用 Windows 约定列出目录路径，包括驱动器冒号和作为目录分隔符的反斜线。例如，

```
MobiLink\redirector
```

在 Unix、Linux 和 Mac OS X 上，您应改用正斜线。例如，

```
MobiLink/redirector
```

如果在多平台环境中使用 SQL Anywhere，则必须注意平台间的路径名区别。

- ◆ **可执行文件** 文档使用 Windows 约定（使用后缀 *.exe*）显示可执行文件名。在 Unix、Linux 和 Mac OS X 上，可执行文件名没有后缀。在 NetWare 上，可执行文件名使用后缀 *.nlm*。

例如，在 Windows 上，网络数据库服务器是 *dbsrv10.exe*。在 Unix、Linux 和 Mac OS X 上，它是 *dbsrv10*。而在 NetWare 上，它是 *dbsrv10.nlm*。

- ◆ **install-dir** 安装过程可让您选择安装 SQL Anywhere 的位置，且文档按约定 *install-dir* 引用此位置。

完成安装后，环境变量 *SQLANY10* 指定包含 SQL Anywhere 组件的安装目录位置 (*install-dir*)。*SQLANYSH10* 指定包含 SQL Anywhere 与其它 Sybase 应用程序共享的组件的目录位置。

有关不同操作系统的 *install-dir* 缺省位置的详细信息，请参见“[SQLANY10 环境变量](#)”一节《[SQL Anywhere 服务器 - 数据库管理](#)》。

- ◆ **samples-dir** 安装过程可让您选择安装 SQL Anywhere 随附示例的位置，且文档按约定 *samples-dir* 引用此位置。

完成安装后，环境变量 *SQLANYSAMP10* 指定包含示例的目录位置 (*samples-dir*)。在 Windows 的 [开始] 菜单中，选择 [程序] ► [SQL Anywhere 10] ► [示例应用程序和项目]，会在此目录中打开一个 Windows 资源管理器窗口。

有关不同操作系统的 *samples-dir* 缺省位置的详细信息，请参见“[示例目录](#)”一节《[SQL Anywhere 服务器 - 数据库管理](#)》。

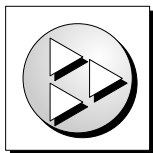
- ◆ **环境变量** 文档介绍设置环境变量。在 Windows 上，使用语法 *%envvar%* 引用环境变量。在 Unix、Linux 和 Mac OS X 上，使用语法 *\$envvar* 或 *\${envvar}* 引用环境变量。

Unix、Linux 和 Mac OS X 环境变量均存储在 shell 和登录启动文件中，如 *.cshrc* 或 *.tcshrc*。

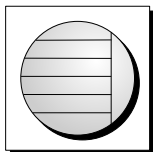
## 图标

本文档中使用了下列图标。

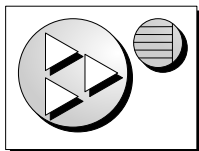
- ◆ 客户端应用程序。



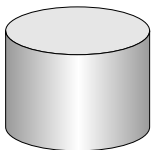
- ◆ 数据库服务器，如 SQL Anywhere。



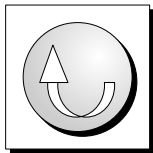
- ◆ UltraLite 应用程序。



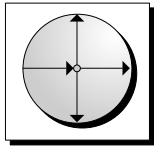
- ◆ 数据库。在某些高水平的图中，可以使用此图标表示数据库和管理该数据库的数据库服务器。



- ◆ 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。



- ◆ Sybase 复制服务器



- ◆ 编程接口。



## 查找详细信息并提供反馈

### 查找详细信息

附加信息和资源（包括代码交流）可从 iAnywhere Developer Network 获得，网址是 <http://www.ianywhere.com/developer/>。

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。您可以在命令提示符下键入 `dbeng10 -v` 来得到这些信息。

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](mailto:sybase.public.sqlanywhere.general)
- ◆ [sybase.public.sqlanywhere.linux](mailto:sybase.public.sqlanywhere.linux)
- ◆ [sybase.public.sqlanywhere.mobilink](mailto:sybase.public.sqlanywhere.mobilink)
- ◆ [sybase.public.sqlanywhere.product\\_futures\\_discussion](mailto:sybase.public.sqlanywhere.product_futures_discussion)
- ◆ [sybase.public.sqlanywhere.replication](mailto:sybase.public.sqlanywhere.replication)
- ◆ [sybase.public.sqlanywhere.ultralite](mailto:sybase.public.sqlanywhere.ultralite)
- ◆ [ianywhere.public.sqlanywhere.qanywhere](mailto:ianywhere.public.sqlanywhere.qanywhere)

#### 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。

### 反馈

我们欢迎您就本文档提出意见、建议或其它反馈信息。

您可以将意见和建议通过电子邮件发送到 SQL Anywhere 文档小组，地址为 [iasdoc@ianywhere.com](mailto:iasdoc@ianywhere.com)。虽然我们对发送到该地址的电子邮件不进行回复，但我们会认真阅读所有建议。

此外，您还可以通过上面所列的新闻组就文档和软件提供反馈信息。

# 第 I 部分 使用 MobiLink 技术

本部分介绍 MobiLink 同步技术并说明如何利用该技术在两个或多个数据库之间复制数据。



---

## 第 1 章

# MobiLink 同步简介

## 目录

MobiLink 快速入门 .....	4
同步系统的组成部分 .....	7
中央数据源 .....	9
MobiLink 客户端 .....	10
MobiLink 服务器 .....	11
同步过程 .....	12
用于编写同步逻辑的选项 .....	18
安全 .....	20
在 Mac OS X 上运行 MobiLink .....	21

## MobiLink 快速入门

MobiLink 的设计用途是在间歇性地与一个或多个中央数据源连接的许多个远程应用程序之间同步数据。在基本 MobiLink 应用程序中，您的远程客户端是 SQL Anywhere 或 UltraLite 数据库，而中央数据源是所支持的兼容 ODBC 的关系数据库之一（SQL Anywhere、Adaptive Server Enterprise、Oracle、Microsoft SQL Server 或 IBM DB2）。此体系结构可以使用 MobiLink 服务器 API 进行扩展，使得实际上在服务器端对于同步目标没有任何限制。

在所有 MobiLink 应用程序中，MobiLink 服务器是同步过程的关键。同步通常是在 MobiLink 远程站点连接到 MobiLink 服务器时开始的。同步期间，远程站点的 MobiLink 客户端可以上载自上一次同步以来对远程数据库所做的更改。MobiLink 服务器收到这些数据时即会更新统一数据库，然后便可将统一数据库中的更改下载到远程数据库。

开始开发 MobiLink 应用程序的最快速方式是使用 [创建同步模型向导]。请参见“[MobiLink 模型简介](#)”一节第 24 页。

### 入门阅读

- ◆ “同步系统的组成部分” 一节第 7 页
- ◆ “同步技术” 《MobiLink - 服务器管理》
- ◆ “数据交换技术概述” 《SQL Anywhere 10 - 简介》

### 其它入门资源

MobiLink 提供了许多示例可供您分析和运行，以探索 MobiLink 的功能。MobiLink 示例随产品安装在 `samples-dir\MobiLink` 中。（有关 `samples-dir` 的位置，请参见“[示例目录](#)”一节《SQL Anywhere 服务器 - 数据库管理》。）

MobiLink 代码交换示例位于 <http://ianywhere.codexchange.sybase.com/servlets/ProjectDocumentList?folderID=319>。

此外，文档中还有一些教程：

- ◆ “探讨 MobiLink 的 CustDB 示例” 第 45 页
- ◆ “探讨 UltraLite 的 CustDB 示例” 《UltraLite - 数据库管理和参考》
- ◆ “研究 MobiLink Contact 示例” 第 67 页
- ◆ “教程：将 MobiLink 用于 Oracle 10g 统一数据库” 第 81 页
- ◆ “教程：使用 Java 同步逻辑” 第 91 页
- ◆ “教程：使用 .NET 同步逻辑” 第 103 页
- ◆ “教程：使用 .NET 和 Java 进行自定义验证” 第 117 页
- ◆ “教程：使用直接行处理” 第 127 页

## MobiLink 特色

MobiLink 同步具有适应性和灵活性。以下是它的一些重要特色：

## 特色

- ◆ **易于入门** 通过使用 [创建同步模型向导]，可以快速地创建同步应用程序。该向导可以处理复杂同步系统的许多困难的实现细节。Sybase Central [模型] 模式允许您脱机查看同步模型，它提供了用于进行更改的简易界面，并提供了供您将模型部署到统一数据库的部署选项。
- ◆ **监控和报告** MobiLink 提供了两种同步监控机制：MobiLink 监控器和统计脚本。
- ◆ **性能调优** 有多种用于调优 MobiLink 性能的机制。例如，您可以调整争用程度、上载高速缓存大小、数据库连接数、记录详细程度或 BLOB 高速缓存大小。
- ◆ **可伸缩性** MobiLink 是一个具有极高可伸缩性和强健性的同步平台。一个 MobiLink 服务器便可以处理数千个并发同步，而通过使用负载平衡可以同时运行多个 MobiLink 服务器。MobiLink 服务器是多线程的，对统一数据库使用连接池。
- ◆ **安全** MobiLink 提供了大量安全性选项，其中包括可与现有验证集成的用户验证、加密以及通过交换安全证书发挥作用的传送层安全性。MobiLink 还提供了经 FIPS 认证的安全性选项。

## 体系结构

- ◆ **数据协调** MobiLink 让您可以选择数据的特定部分进行同步。MobiLink 同步还允许您解决在不同数据库中所做更改之间的冲突。同步过程由同步逻辑控制，可以将该逻辑编写为 SQL、Java 或 .NET 应用程序。同步逻辑的每个部分称为一个**脚本**。例如，您可以使用脚本指定如何将上载的数据应用到统一数据库、指定下载的内容以及处理统一数据库与远程数据库之间的不同模式和名称。基于事件的脚本编写方式为同步过程的设计提供了极大的灵活性，可以设计出冲突解决、错误报告和用户验证等功能。
- ◆ **双向同步** 可以在任何位置对数据库进行更改。
- ◆ **仅上载同步或仅下载同步** 可以选择仅执行上载或仅执行下载，也可以选择执行双向同步。
- ◆ **基于文件的下载** 可以文件形式分发下载，从而实现同步更改的脱机分发。此功能包括用于确保应用了正确数据的功能。
- ◆ **服务器启动的同步** 可以从统一数据库启动 MobiLink 同步。这意味着您可以将数据更新推送到远程数据库，也可以让远程数据库将数据上载到统一数据库。
- ◆ **选择网络协议** 同步可以通过 TCP/IP、HTTP 或 HTTPS 执行。Palm 设备可以通过 HotSync 进行同步。Windows CE 设备可以使用 ActiveSync 进行同步。
- ◆ **基于会话** 所有更改都可以通过单个事务进行上载和下载。每次同步成功完成后，统一数据库和远程数据库就会变得一致。（如果希望保留事务的顺序，还可以选择将远程数据库上的每个事务作为单独的事务上载。）

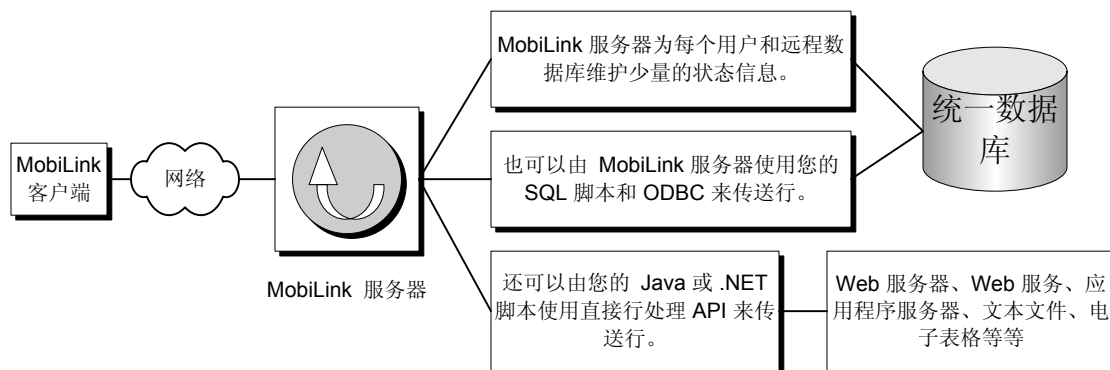
要么同步整个事务，要么不同步事务的任何部分。这样可以确保每个数据库的事务完整性。

- ◆ **数据一致性** MobiLink 使用松散一致性策略来运行。每过一段时间便会以一致的方式与每个站点同步所有更改，但在任何时刻不同的站点所拥有的数据副本可能不同。
- ◆ **多种多样的硬件和软件平台** 多种得到广泛使用的数据库管理系统（SQL Anywhere、Adaptive Server Enterprise、Oracle、Microsoft SQL Server 或 IBM DB2 UDB）都可用作 MobiLink 统一数据库，也可以使用 MobiLink 服务器 API 定义到任意数据源的同步。远程数据库可以是 SQL Anywhere 或 UltraLite。MobiLink 服务器可在 Windows 或 UNIX/Linux 平台上运行。SQL

Anywhere 可在 Windows、Windows CE 或 UNIX 上运行。UltraLite 可在 Palm 或 Windows CE 上运行。

## 同步系统的组成部分

在 MobiLink 同步中，许多客户端通过 MobiLink 服务器与中央数据源同步。



- ◆ **MobiLink 客户端** 可将客户端安装在手持式设备（如 Palm Pilot 或者 Windows Mobile 设备）、服务器、台式计算机或者智能电话上。共支持两种类型的客户端：UltraLite 和 SQL Anywhere 数据库。在一个 MobiLink 安装中可以同时使用上述两种或其中一种客户端。

请参见“[MobiLink 客户端](#)”一节第 10 页。

- ◆ **网络** MobiLink 服务器和 MobiLink 客户端之间的连接可使用多个协议。

有关详细信息，请参见：

- ◆ **MobiLink 服务器**：“[-x 选项](#)”一节 《[MobiLink - 服务器管理](#)》
- ◆ **UltraLite 和 SQL Anywhere 客户端**：“[MobiLink 客户端网络协议选项](#)” 《[MobiLink - 客户端管理](#)》
- ◆ **MobiLink 服务器** 此服务器管理同步过程并提供所有 MobiLink 客户端与统一数据库服务器之间的接口。

请参见“[MobiLink 服务器](#)” 《[MobiLink - 服务器管理](#)》。

- ◆ **统一数据库** 此数据库包含 MobiLink 同步所需的系统表和过程，以及同步所需的系统信息。此数据库通常还包括同步系统中信息的集中副本。此数据库可以是 SQL Anywhere、Adaptive Server Enterprise、Oracle、DB2 或 Microsoft SQL Server。

请参见“[中央数据源](#)”一节第 9 页和“[MobiLink 统一数据库](#)” 《[MobiLink - 服务器管理](#)》。

- ◆ **状态信息** MobiLink 服务器必须维护统一数据库中系统表内的一些信息。这通过 ODBC 连接来实现。
- ◆ **SQL 行处理** 如果为 MobiLink 服务器提供 SQL 脚本，则它将使用这些脚本，并通过 ODBC 连接在 MobiLink 服务器和统一数据库之间传输行。

请参见“用于编写同步逻辑的选项”一节第 18 页和“用于 MobiLink 的 iAnywhere Solutions ODBC 驱动程序”《MobiLink - 服务器管理》。

- ◆ **直接行处理** 除了统一数据库，您还可以选择使用 MobiLink 直接行处理 API 与其它数据源进行同步。这些连接可能会使用多种接口。

请参见“直接行处理”《MobiLink - 服务器管理》。

您可以为远程数据库中的每个表编写**同步脚本**，并将这些脚本保存在统一数据库中的 MobiLink 系统表内。这些脚本确定如何处理上载数据以及下载哪些数据。有两类脚本：表脚本和连接级脚本。请参见：

- ◆ “MobiLink 事件概述”一节《MobiLink - 服务器管理》
- ◆ “编写同步脚本”《MobiLink - 服务器管理》
- ◆ “同步事件”《MobiLink - 服务器管理》
- ◆ “用于编写同步逻辑的选项”一节第 18 页

### MobiLink 开发环境

有两种方法可以开发和维护您的 MobiLink 同步系统：

- ◆ **创建模型** 在 Sybase Central MobiLink 插件中，可以使用 [创建同步模型向导] 和 [模型] 模式自动化统一数据库、远程数据库和同步脚本的创建和设置，并运行 MobiLink 服务器和客户端。

请参见“MobiLink 模型简介”一节第 24 页。

- ◆ **直接更改系统对象** MobiLink 提供系统过程和命令行实用程序，您可以使用它们在 MobiLink 系统表中创建数据库对象并注册同步逻辑。请参见：

- ◆ “MobiLink 服务器系统过程”《MobiLink - 服务器管理》
- ◆ “MobiLink 实用程序”《MobiLink - 服务器管理》
- ◆ “MobiLink 服务器系统表”《MobiLink - 服务器管理》

您还可以使用 Sybase Central [管理] 模式直接更改系统对象。请参见“管理模式”一节第 28 页。

## 中央数据源

MobiLink 同步系统可具有两种类型的中央数据源：

- ◆ **统一数据库** 您必须拥有统一数据库。它可以是 SQL Anywhere、Adaptive Server Enterprise、Oracle、Microsoft SQL Server 或 IBM DB2。（MobiLink 支持用于 Linux、Unix 和 Windows 的 DB2，但不支持用于 AS/400 或主机的 DB2。）要为数据库成为统一数据库做好准备，需运行一个安装脚本，该安装脚本安装同步所需的 MobiLink 系统表、存储过程、触发器和事件。MobiLink 系统表保存同步所需的状态、配置和用户信息。在大多数应用程序中，统一数据库也是同步系统中信息的主存储库。

请参见“[MobiLink 统一数据库](#)”《[MobiLink - 服务器管理](#)》。

- ◆ **另一个中央数据源** 或者，可以将全部或部分中央数据存储于统一数据库以外的其它数据源内。举例来说，此数据源可以是应用程序服务器、电子表格、Web 服务器、Web 服务或文本文件。可使用 MobiLink 直接行处理 API 来读取上载并创建下载。可以使用 Java 或 .NET 脚本编写同步逻辑。

请参见“[直接行处理](#)”《[MobiLink - 服务器管理](#)》。

## MobiLink 客户端

每个远程数据库连同其应用程序一起被称为一个 **MobiLink 客户端**。共支持两种类型的 MobiLink 客户端：

- ◆ SQL Anywhere
- ◆ UltraLite

请参见“[MobiLink 客户端介绍](#)” 《[MobiLink - 客户端管理](#)》。

远程数据库可包含与中央数据源相同的表，或包含一个子集，或包含完全不同的模式。

有关如何处理不同模式的信息，请参见“[在远程数据库之间对行进行分区](#)”一节 《[MobiLink - 服务器管理](#)》。

## MobiLink 服务器

MobiLink 服务器 (mlsrv10) 位于 MobiLink 客户端和中央数据源之间，并且客户端和服务器之间的所有通信都通过 MobiLink 服务器进行。

有关如何运行 mlsrv10 的详细信息，请参见：

- ◆ “MobiLink 服务器” 《MobiLink - 服务器管理》
- ◆ “MobiLink 服务器选项” 《MobiLink - 服务器管理》

## 同步过程

**同步**是 MobiLink 客户端和同步服务器之间的数据交换过程。在此过程中，客户端必须建立并维护与同步服务器的会话。如果成功，此会话将使远程数据库和统一数据库保持相互一致的状态。

一般由客户端开始进行同步过程。此过程通过与 MobiLink 服务器建立连接开始。

### 上载和下载

对于行的上载，MobiLink 客户端准备并发送一个**上载**，它包含从上次同步后在 MobiLink 客户端上经过更新、插入或删除的所有行的列表。类似地，对于行的下载，MobiLink 服务器准备并发送一个**下载**，它包含插入、更新和删除的列表。

1. **上载** 缺省情况下，MobiLink 客户端会自动跟踪上次成功同步后远程数据库中插入、更新和删除了哪些行。连接一旦建立，MobiLink 客户端将一个列出所有这些更改的列表上载到同步服务器。

上载由远程数据库中被修改行的一组新行值和旧行值组成。（更新具有新行值和旧行值；删除具有旧行值；插入具有新行值。）如果某行被更新或删除，旧行值是指就在上次成功同步后所存在的那些值。如果某一行被插入或更新，新行值是指当前的行值。即使在到达当前状态之前，行已被修改过数次，也不会发送任何中间值。

MobiLink 服务器接收上载并执行您定义的上载脚本。缺省情况下，所有的更改都会在一次事务中完成应用。此后，MobiLink 服务器将提交该事务。

#### 注意

MobiLink 运行时使用 ODBC 隔离级别 `SQL_TXN_READ_COMMITTED` 作为统一数据库的缺省隔离级别。如果用于统一数据库的 RDBMS 支持快照隔离，并且为数据库启用了快照，则缺省情况下 MobiLink 对下载使用快照隔离。请参见“[MobiLink 隔离级别](#)”一节《[MobiLink - 服务器管理](#)》。

2. **下载** MobiLink 服务器将使用您创建的同步逻辑，编译要在 MobiLink 客户端上进行插入、更新或者删除的行的列表。它将这些行下载到 MobiLink 客户端。为了对此列表进行编译，MobiLink 服务器将在统一数据库中打开一个新事务。

MobiLink 客户端接收下载。当下载到到达时，MobiLink 客户端认为统一数据库已成功应用所有上载的更改。并确保这些更改不会再发送到统一数据库中。

接下来，MobiLink 客户端将自动对下载进行处理，删除旧行、插入新行并更新已更改的行。所有的更改将在远程数据库的一个事务中完成。此后，MobiLink 客户端提交该事务。

在 MobiLink 同步过程中，只有很少的明显交换信息。客户端建立并上载整个上载。作为响应，同步服务器建立并下载整个下载。在通信较慢而等待时间较长的时候，例如在使用电话线或公共无线网时，对协议中的无用信息进行限制非常重要。

### 另请参见

- ◆ “[MobiLink 事件概述](#)”一节《[MobiLink - 服务器管理](#)》
- ◆ “[上载过程中的事件](#)”一节《[MobiLink - 服务器管理](#)》
- ◆ “[下载过程中的事件](#)”一节《[MobiLink - 服务器管理](#)》

## MobiLink 事件

当 MobiLink 客户端开始进行同步时，将发生多个同步事件。在发生事件时，MobiLink 将查找与同步事件匹配的脚本。此脚本包含一些说明，详细指出您需要服务器完成的操作。如果您已为事件定义了脚本且将脚本放入 MobiLink 系统表中，则脚本被调用。

有关同步事件和脚本的详细信息，请参见：

- ◆ “编写同步脚本” 《MobiLink - 服务器管理》
- ◆ “同步事件” 《MobiLink - 服务器管理》

## MobiLink 脚本

每次发生事件时，MobiLink 服务器都将执行关联的脚本（如果您已创建了一个关联脚本）。如果脚本不存在，将发生序列中的下一个事件。

### 注意

使用 [创建同步模式向导] 创建 MobiLink 应用程序时，会为您创建所有需要的 MobiLink 脚本。然而，在某些情况下，您可能需要编辑它们。

以下是表的典型上传脚本：

事件	示例脚本内容
upload_insert	<pre>INSERT INTO emp (emp_id,emp_name) VALUES {ml r.emp_id}, {ml r.emp_name}</pre>
upload_delete	<pre>DELETE FROM emp WHERE emp_id = {ml r.emp_id}</pre>
upload_update	<pre>UPDATE emp SET emp_name = {ml r.emp_name} WHERE emp_id = {ml r.emp_id}</pre>

第一个事件 upload\_insert 触发 upload\_insert 脚本的运行，该脚本将 emp\_id 和 emp\_name 插入到 emp 表中。同样地，upload\_delete 和 upload\_update 脚本将对同一 emp 表的删除和更新操作执行类似的功能。

下载脚本使用游标。以下是一个 download\_cursor 脚本的示例：

```
SELECT order_id, cust_id
FROM ULOrder
WHERE last_modified >= {ml s.last_table_download}
AND emp_name = {ml r.emp_id}
```

在您的 SQL 同步脚本中或从您的 SQL 同步脚本调用的过程或触发器中，不应有任何隐式或显式提交或回退。SQL 脚本内的 COMMIT 或 ROLLBACK 语句会改变同步步骤的事务性质。如果您使用这两个语句，则在出现故障时 MobiLink 将无法保证数据的完整性。

## 您可以使用 SQL、Java 或 .NET 编写脚本

可以使用统一数据库的本地 SQL 方言或使用 Java 或 .NET 同步逻辑编写脚本。Java 和 .NET 同步逻辑可用于编写由 MobiLink 服务器激活的代码，以连接到数据库、操纵变量、直接操纵上载行操作，或者向下载添加行操作。有面向 Java 的 MobiLink 服务器 API 和面向 .NET 的 MobiLink 服务器 API，它们提供满足同步需要的类和方法。

有关同步逻辑编程的详细信息，请参见“用于编写同步逻辑的选项”一节第 18 页。

有关 RDBMS 相关的脚本编写的信息，例如为 Oracle、Microsoft SQL Server、IBM DB2 UDB 或 Adaptive Server Enterprise 数据库编写脚本，请参见“MobiLink 统一数据库”《MobiLink - 服务器管理》。

## 存储脚本

SQL 脚本存储在统一数据库的 MobiLink 系统表中。对于使用 MobiLink 服务器 API 编写的脚本，您可将完全限定的方法名作为脚本存储。可以使用多种方法将脚本添加到统一数据库：

- ◆ 如果使用 [创建同步模式向导]，则在部署项目时会将脚本存储在 MobiLink 系统表中。
- ◆ 可以使用建立统一数据库时安装的存储过程，手工向系统表添加脚本。
- ◆ 可以使用 Sybase Central 手工向系统表添加脚本。

请参见“添加和删除脚本”一节《MobiLink - 服务器管理》。

## 同步过程中的事务

MobiLink 服务器在一次事务中将各个 MobiLink 客户端上载的更改并入统一数据库中。它在插入新行、删除旧行、更新以及解决任何冲突后，便会提交这些更改。

MobiLink 服务器使用另一个事务准备下载，其中包括所有的删除、插入以及更新。如果您指定下载确认且客户端确认下载成功，则 MobiLink 服务器将提交下载事务。如果指定了下载确认后，应用程序遇到问题或无法应答，则 MobiLink 服务器将回退下载事务。缺省情况下，不使用下载确认。

### **不要在脚本内提交或回退事务**

脚本内的 COMMIT 或 ROLLBACK 语句会改变同步步骤的事务性质。如果您使用这两个语句，则在出现故障时 MobiLink 将无法保证数据的完整性。在您的同步脚本中或从您的同步脚本调用的过程或触发器中，不应有任何隐式或显式提交或回退。

## 跟踪下载信息

MobiLink 使用上次下载时间戳（存储在远程数据库中），帮助简化创建下载的过程。

下载事务的主要作用是选择统一数据库中的行。如果下载失败，则远程数据库将重新上载与上次下载时间戳相同的时间戳，这样不会丢失数据。

请参见“在脚本中使用上次下载时间”一节《MobiLink - 服务器管理》。

## 开始和结束事务

MobiLink 客户端在一个事务中处理下载中的信息。对行进行插入、更新及删除操作，通过与统一数据库的交互来使远程数据库处于最新状态。

MobiLink 服务器还使用其它两个事务，一个是在同步开始时，另一个是在结束时。这些事务使您可以记录有关每次同步及其持续时间的信息。这样，您可以记录有关同步尝试、同步成功以及同步持续时间的统计。因为数据是在过程中的各个点处提交，所以这些事务也可以使您在分析失败的同步尝试时可以提交有用的数据。

## 另请参见

- ◆ “MobiLink 事件概述”一节 《MobiLink - 服务器管理》
- ◆ “上载过程中的事件”一节 《MobiLink - 服务器管理》
- ◆ “下载过程中的事件”一节 《MobiLink - 服务器管理》

## 如何处理同步失败

MobiLink 同步具有容错功能。例如，如果通信链接在同步过程中失败，远程数据库和统一数据库的状态将保持一致。

在客户端，故障由一个返回代码指示。

同步失败的处理方式依发生时间的不同而不同。以下情况采用不同的方式进行处理：

- ◆ **上载中的故障** 如果在创建或应用上载时出现故障，远程数据库将保持与同步开始时完全相同的状态。在服务器端，任何已应用的上载部分将被回退。
- ◆ **上载和下载之间的故障** 如果故障在上载完成之后、MobiLink 客户端接收到下载之前发生，客户端将无法确定上载更改是否成功地应用到统一数据库中。上载可能已被全部应用并提交，故障也可能出现在服务器应用整个上载之前。此时 MobiLink 服务器会自动回退统一数据库中的未完成事务。

MobiLink 客户端维护所有上载更改的记录，以备这些更改必须再次发送。下一次客户端同步时，它会在创建新的上载之前请求上一个上载的状态。如果没有提交上一个上载，则新的上载将包含自上一个上载时起发生的所有更改。

- ◆ **下载中的故障** 如果指定了下载确认且在应用下载时故障发生于远程设备，则所有已应用的下载部分都将回退，而远程数据库将与下载前保持相同的状态。MobiLink 服务器也会回退统一数据库中的下载事务。

在所有故障可能发生的情况中，都不会丢失数据。这将由 MobiLink 服务器与 MobiLink 客户端为您进行管理。开发人员或用户不必担忧在其应用程序中对数据一致性进行维护。

## 如何处理上载

当 MobiLink 服务器接收到来自 MobiLink 客户端的上载时，只有在同步完成之后才存储整个上载。这样做有以下原因：

- ◆ **过滤下载行** 确定下载行的最常用技术是下载最近一次下载后修改过的行。在同步的时候，上载先于下载进行。在上载中插入或更新的任何行都将是上一次下载后修改过的行。

编写一个 `download_cursor` 脚本从下载中去掉那些作为上载部分发送的行是比较困难的。因此，MobiLink 服务器将自动从下载中删除这些行。

- ◆ **处理插入和更新** 缺省情况下，上载中的表以避免参照完整性违规的顺序应用于统一数据库中。上载中的表将根据外键关系排序。例如，如果表 A 和表 C 都有外键引用了表 B 中的主键列，那么首先上载表 B 行的插入和更新。
- ◆ **在插入和更新之后处理删除** 删除操作在应用了所有插入和更新之后应用到统一数据库。当应用删除操作时，将以与其在上载中出现方式相反的顺序处理表。当一个被删除的行引用另一个表中也被删除的行时，这一操作顺序将保证引用行在被引用行之前删除。
- ◆ **死锁**  
当上载应用于统一数据库时，可能会因为与其它事务并发而出现死锁。这些事务可能是来自其它 MobiLink 服务器数据库连接的上载事务，或者是来自其它使用统一数据库的应用程序事务。上载事务发生死锁时将被回退，而 MobiLink 服务器将自动重新开始应用上载。

### 性能提示

编写同步脚本时应尽可能地避免争用。在多个用户同时进行同步时，争用将对性能有很大影响。

## 参照完整性与同步

所有 MobiLink 客户端在把下载并入远程数据库时将确保参照完整性。

缺省情况下，MobiLink 客户端自动删除所有违反参照完整性的行，防止下载事务出现故障。

此功能包括以下优点：

- ◆ 同步脚本中的差错保护。由于脚本的灵活性，有可能意外地将破坏远程数据库完整性的行下载下来。MobiLink 客户端将对参照完整性进行无干预自动维护。
- ◆ 您可以使用该参照完整性机制有效地从远程数据库中删除信息。MobiLink 客户端只把删除发送给父记录，由此为您自动清除所有的子记录。这将大大降低 MobiLink 须发送给远程数据库的通信量。

如果 MobiLink 客户端必须显式删除行以维护参照完整性，它会提供通知。

- ◆ 对于 SQL Anywhere 客户端，`dbmsync` 将在日志中写入一个条目。同时还有 `dbmsync` 事件挂接供您使用。请参见：
  - ◆ [“sp\\_hook\\_dbmsync\\_download\\_ri\\_violation”](#) 一节 《MobiLink - 客户端管理》
  - ◆ [“sp\\_hook\\_dbmsync\\_download\\_log\\_ri\\_violation”](#) 一节 《MobiLink - 客户端管理》
- ◆ 对于 UltraLite 客户端，将引发 `SQL_ROW_DELETED_TO_MAINTAIN_REFERENTIAL_INTEGRITY` 警告。此警告以表名为参数。在为了维护参照完整性而删除每行时都将引发此警告。如果您希望同步仍然继续，应用

程序可以忽略警告。如果希望显式处理警告，可以使用错误回调函数来捕获它们，例如，计算删除的行数。

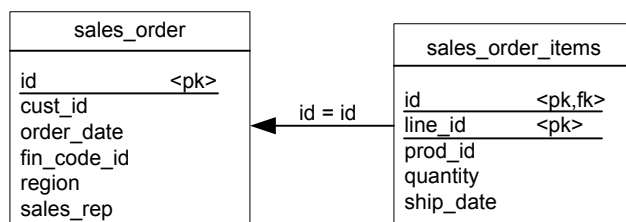
在引发警告时，如果要放弃同步，则必须实现同步观察器，然后从错误回调函数给观察器发信号（也许通过一个全局变量）。这种情况下，将会在下次调用观察器时放弃同步。

### 在事务结束时检查参照完整性

MobiLink 客户端在单个事务中并入来自下载的更改。为了提供更大的灵活性，参照完整性检查将在此事务结束处进行。由于检查被延迟，因此数据库可能会暂时出现违反参照完整性的状态。这是因为违反参照完整性的行在提交下载之前将自动被删除。

### 示例

假设 UltraLite 销售应用程序除其它表之外包含以下两个表。一个表包含销售订单。另一张表是各订单中销售的项目。它们具有以下关系：



如果您使用 `sales_order` 表的 `download_delete_cursor` 删除一个订单，缺省参照完整性机制将自动地删除 `sales_order_items` 表中所有指向被删除销售订单的行。

这种机制有以下优点：

- ◆ 您不再需要 `sales_order_items` 脚本，因为该表中的行将被自动删除。
- ◆ 提高了同步的效率。您不需要下载从 `sales_order_item` 表中删除的行。如果每个销售订单包含很多项目，由于现在减少了下载，所以性能得以提高。此技术在使用较慢的通信方法时尤其适用。

### 更改缺省行为

对于 SQL Anywhere 客户端，可以使用 `sp_hook_dbmsync_download_ri_violation` 客户端事件挂接来处理参照完整性违规。Dbmsync 也会在其日志中写入一个条目。

请参见：

- ◆ [“sp\\_hook\\_dbmsync\\_download\\_log\\_ri\\_violation”](#) 一节 《MobiLink - 客户端管理》
- ◆ [“sp\\_hook\\_dbmsync\\_download\\_ri\\_violation”](#) 一节 《MobiLink - 客户端管理》

## 用于编写同步逻辑的选项

可以使用 SQL、Java（使用面向 Java 的 MobiLink 服务器 API）或者 .NET（使用面向 .NET 的 MobiLink 服务器 API）来编写 MobiLink 同步脚本。

当同步到支持的统一数据库（SQL Anywhere、Adaptive Server Enterprise、Oracle、SQL Server 或 DB2）时，SQL 同步逻辑通常是最佳方法。

在对所支持的统一数据库之外的其它数据源进行同步时，Java 或 .NET 是非常有用的。在 SQL 语言的局限性或者数据库管理系统的功能限制了您的设计，或者您只是需要跨越不同 RDBMS 类型的可移植性时，Java 或 .NET 也是会非常有用的。

Java 和 .NET 同步逻辑可以实现与 SQL 逻辑完全相同的功能。正如 MobiLink 服务器可以在 MobiLink 事件发生时访问 SQL 脚本一样，它也可以在 MobiLink 事件发生时调用 Java 或 .NET 方法。当使用 Java 或 .NET 工作时，可以使用事件进行一些额外处理，但当处理那些直接处理上载或下载行的事件的脚本时，您的实现必须返回 SQL 字符串。除了用于直接行处理的两个事件外，不能直接通过 Java 或 .NET 同步逻辑访问上载和下载：MobiLink 将 Java 或 .NET 返回的字符串作为 SQL 执行。

直接行处理（使用事件 `handle_UploadData` 和 `handle_DownloadData` 与除了统一数据库外的其它数据源同步）*确实* 直接操纵上载和下载行——而不使用统一数据库。

以下是您可能需要考虑使用 Java 或 .NET 编写脚本的一些情形：

- ◆ **直接行处理** 利用 Java 和 .NET 同步逻辑，您可以使用 MobiLink 来访问除了统一数据库外的其它数据源（如应用程序服务器、Web 服务器和文件）中的数据。
- ◆ **验证** 可以使用 Java 或 .NET 编写一个用户验证过程，因此 MobiLink 验证会与您的公司安全策略相集成。
- ◆ **存储过程** 如果您的数据库不具备生成用户定义的存储过程的能力，您可以使用 Java 或 .NET 创建一个方法来执行所需的功能。
- ◆ **外部调用** 如果程序需要在同步事件中联系外部服务器，则您可以使用 Java 或 .NET 同步逻辑执行由同步事件触发的操作。可以在多个连接之间共享 Java 和 .NET 同步逻辑。
- ◆ **变量** 如果您的数据库缺乏处理变量的能力，您可以使用 Java 或 .NET 创建一个持续贯穿于整个连接或同步过程的变量。（或者使用 SQL 脚本，您可以使用用于所有统一数据库类型的用户定义命名参数。请参见“[用户定义的命名参数](#)”一节《[MobiLink - 服务器管理](#)》。）

### MobiLink 服务器 API

可通过 MobiLink 服务器 API 使用 Java 和 .NET 同步逻辑。MobiLink 服务器 API 是几组用于 MobiLink 同步的类和接口。

面向 Java 的 MobiLink 服务器 API 可以帮助您：

- ◆ 像访问 JDBC 连接一样访问到统一数据库的现有 ODBC 连接。
- ◆ 使用诸如 JDBC、Web 服务和 JNI 这样的接口访问替代数据源。

- ◆ 可以创建到统一数据库的新 JDBC 连接，使数据库在当前同步连接之外进行更改。例如，即使同步连接执行回退，您也可以使用它来进行错误记录或者审计。
- ◆ 对于与统一数据库之间的同步，可以在 MobiLink 服务器执行它之前编写和调试 Java 代码。用于许多数据库管理系统的 SQL 的开发环境同可用于 Java 应用程序的环境相比显得相对简单。
- ◆ SQL 行处理和直接行处理。
- ◆ 完整丰富的 Java 语言及其大量的现有代码和库。

请参见“用于 Java 的 MobiLink 服务器 API 参考”一节《MobiLink - 服务器管理》。

面向 .NET 的 MobiLink 服务器 API 可以帮助您：

- ◆ 使用从 .NET 调用 ODBC 的 iAnywhere 类访问到统一数据库的现有 ODBC 连接。
- ◆ 使用诸如 ADO.NET、Web 服务和 OLE DB 这样的接口访问替代数据源。
- ◆ 对于与统一数据库之间的同步，可以在 MobiLink 服务器执行它之前编写和调试 .NET 代码。用于许多数据库管理系统的 SQL 的开发环境同可用于 .NET 应用程序的环境相比显得相对简单。
- ◆ SQL 行处理和直接行处理。
- ◆ 在 .NET 公共语言运行库 (CLR) 中运行的代码，这些代码允许访问所有 .NET 库，包括 SQL 行处理和直接行处理。

请参见“用于 .NET 参考的 MobiLink 服务器 API”一节《MobiLink - 服务器管理》。

### 另请参见

- ◆ “编写同步脚本” 《MobiLink - 服务器管理》
- ◆ “同步技术” 《MobiLink - 服务器管理》
- ◆ “使用 Java 语言编写同步脚本” 《MobiLink - 服务器管理》
- ◆ “使用 .NET 编写同步脚本” 《MobiLink - 服务器管理》
- ◆ “直接行处理” 《MobiLink - 服务器管理》

## 安全

在大范围的分布式系统如 MobiLink 系统中，有关确保数据安全性涉及以下几个方面：

- ◆ **统一数据库中的数据保护** 统一数据库中的数据可以由 DBMS 用户验证系统和其它安全性功能进行保护。

有关详细信息，请参见 DBMS 文档。如果您使用 SQL Anywhere 统一数据库，请参见“[保护数据的安全](#)”《[SQL Anywhere 服务器 - 数据库管理](#)》。

- ◆ **远程数据库中的数据保护** 如果使用的是 SQL Anywhere 远程数据库，则使用 SQL Anywhere 安全性功能对数据进行保护。缺省情况下，这些功能旨在防止通过客户端/服务器通信进行未经授权的访问，但是却无法防止直接从数据库文件中抽取信息这种后果严重的行为。

客户端的文件由客户端操作系统的安全性功能进行保护。

如果使用的是 SQL Anywhere 远程数据库，请参见“[保护数据的安全](#)”《[SQL Anywhere 服务器 - 数据库管理](#)》。

如果使用的是 UltraLite 数据库，请参见“[UltraLite 安全注意事项](#)”一节《[UltraLite - 数据库管理和参考](#)》。

- ◆ **同步中的数据保护** 从 MobiLink 客户端到 MobiLink 服务器进行的通信可以通过 MobiLink 传送层安全性功能进行保护。

请参见“[传送层安全](#)”《[SQL Anywhere 服务器 - 数据库管理](#)》。

- ◆ **保护同步系统不被未授权用户访问** MobiLink 同步可以由基于口令的用户验证系统提供保护。这种机制防止未授权用户进行数据同步。

请参见“[MobiLink 用户](#)”《[MobiLink - 客户端管理](#)》。

## 在 Mac OS X 上运行 MobiLink

为了在 Mac OS X 上同步 MobiLink 统一数据库，您可以将 SQL Anywhere ODBC 驱动程序用作驱动程序管理器。请参见“在 Mac OS X 上创建 ODBC 数据源”一节《SQL Anywhere 服务器 - 数据库管理》。

### ◆ 在 Mac OS X 上启动 MobiLink 服务器

1. 启动 SyncConsole。

在 Finder 中，双击 [SyncConsole]。SyncConsole 应用程序位于 */Applications/SQLAnywhere10* 中。

2. 选择 [文件] ► [新建] ► [MobiLink 服务器]。

即会出现服务器选项对话框。

3. 配置 MobiLink 服务器：

- a. 在 [连接参数] 字段中，输入以下字符串：

```
dsn=dsn-name
```

*dsn-name* 是 SQL Anywhere ODBC 数据源名。有关创建 ODBC 数据源的信息，请参见“在 Unix 和 Mac OS X 上设置环境变量”一节《SQL Anywhere 服务器 - 数据库管理》。

如果 *dsn-name* 有空格，则使用双引号将字符串引起来。例如：

```
dsn="SQL Anywhere 10 Demo"
```

- b. 将 [选项] 字段留空。

[选项] 字段允许您控制 MobiLink 服务器行为的许多方面。有关选项的完整列表，请参见“mlsrv10 语法”一节《MobiLink - 服务器管理》。

4. 启动 MobiLink 服务器。

单击 [启动] 以启动服务器。即会出现 [服务器消息] 窗口并显示消息，表示服务器已准备就绪可以接受同步请求了。

### ◆ 在 Mac OS X 上启动 dbmlsync

1. 启动 SyncConsole。

在 Finder 中，双击 [SyncConsole]。SyncConsole 应用程序位于 */Applications/SQLAnywhere10* 中。

2. 选择 [文件] ► [新建] ► [MobiLink 客户端]。

即会出现客户端选项对话框。

选项对话框具有许多配置选项，这些配置选项对应于 dbmlsync 命令行选项。有关完整列表，请参见“dbmlsync 语法”一节《MobiLink - 客户端管理》。

[登录]、[数据库]、[网络]和[高级]选项卡上的选项均定义了从MobiLink客户端到SQL Anywhere远程数据库的连接。许多情况下，只需在[登录]选项卡上指定ODBC数据源即可进行连接。

[DBMLSync]选项卡上的选项定义了到MobiLink服务器的连接的各个方面。如果在远程数据库发布和预订中定义了这些功能，则可将此选项卡上的选项留空。

### ◆ 在 Mac OS X 上运行示例数据库

1. 指定 *sa\_config* 配置脚本的来源。

有关详细信息，请参见“在 Unix 和 Mac OS X 上设置环境变量”一节《SQL Anywhere 服务器 - 数据库管理》。

2. 设置 ODBC 数据源。例如：

```
dbdsn -w "SQL Anywhere 10 Demo"  
-c "uid=DBA;pwd=sql;dbf=/Applications/SQLAnywhere10/System/demo.db"
```

3. 运行 MobiLink 服务器。例如：

```
dbmlsrv10 -c "dsn=SQL Anywhere 10 Demo"
```

---

## 第 2 章

# MobiLink 模型

## 目录

MobiLink 模型简介 .....	24
创建模型 .....	25
模型模式 .....	28
部署模型 .....	38

## MobiLink 模型简介

**同步模型**是一种可用来轻松创建 MobiLink 应用程序的工具。同步模型是一个文件，它由 Sybase Central 中的**创建同步模型向导**创建。

运行 [创建同步模型向导] 时，将提示您连接统一数据库以便获得模式信息。如果尚未将数据库设置为统一数据库，则该向导可应用设置脚本创建同步所需要的 **MobiLink** 系统表和其它对象；除此之外，在您部署模型之前不对统一数据库进行任何更改。向导完成后，到数据库的连接会被关闭。

完成 [创建同步模型向导] 后，模型将以**模型模式**显示。可以使用 [模型] 模式自定义模型。如果在模型模式下，则您以脱机方式工作：不对统一数据库进行任何更改。模型存储在扩展名为 *.mlsm* 的模型文件中。

模型完成后，可以使用**部署同步模型向导**进行部署。[部署同步模型向导] 使用您选择的部署选项创建脚本文件以运行同步服务器和客户端。在部署时您可以选择对现有数据库进行更改，也可以选择让向导创建您自己要运行的文件。

部署之后，可以对模型或数据库做出进一步更改然后重新部署。

## 创建模型

使用 [创建同步模型向导] 创建 MobiLink 模型。

### 创建同步模型向导

[创建同步模型向导] 会引导您逐步完成 MobiLink 同步应用程序的创建。

#### ◆ 使用 [创建同步模型向导] 设置 MobiLink 应用程序的概述

1. 打开 Sybase Central。  
选择 [程序] ► [SQL Anywhere 10] ► [Sybase Central]。
2. 从 [工具] 菜单中，选择 [MobiLink 10] ► [设置 MobiLink 同步]。  
随即出现 [创建同步模型向导]。
3. 选择模型的名称和位置。模型存储在扩展名为 *.mlsm* 的模型文件中。  
*注意：* 该名称和位置用作由向导所创建的文件和目录的缺省名称。
4. [主键需求] 页面即会出现。  
包含此页的目的是为了保证安全。它提醒您保持主键的唯一性和永久性。要继续进行，必须选中三个复选框。通过选择此页面的底部的框可在今后禁用此页。  
请参见“[维护唯一主键](#)”一节《[MobiLink - 服务器管理](#)》。
5. [统一数据库模式] 页面出现。您必须连接到在 MobiLink 应用程序中作为统一数据库的数据库，以便向导获得它的模式信息。  
如果尚未将此数据库设置为统一数据库，向导会提示您进行设置。MobiLink 设置过程会将系统对象添加到 MobiLink 所需的数据库中。如果您选择做出选择，这些对象就会立即添加到统一数据库中。（可以选择稍后在部署向导中或通过自行应用设置文件进行此设置。）  
请参见“[建立统一数据库](#)”一节《[MobiLink - 服务器管理](#)》。  
如果您选择其它的统一数据库或在退出向导时，将关闭与该统一数据库的连接。从这时起，您在此向导中所做的更改即是针对模型文件，而不是统一数据库。
6. [远程数据库模式] 页面出现。可基于统一数据库或现有远程数据库创建您的远程数据库模式。现有远程数据库可以是 SQL Anywhere 或 UltraLite。（部署时，可将此模式应用至新的或现有的远程数据库。请参见“[远程数据库](#)”一节第 26 页。）  
对于新的 SQL Anywhere 远程数据库，远程表的所有者与相应的统一数据库表的所有者相同。如果需要不同的所有者，应改用具备所需的表所有权的现有远程数据库。
7. 请按照向导中的其余说明进行操作。请尽可能使用基于最佳做法提出的缺省建议值。所有页面都有联机帮助。
8. 单击 [完成]。

单击 [完成] 后，刚刚创建的模型即会以模型模式打开。同时，与统一数据库的连接将关闭。您现在以脱机方式工作，且可对模型进行更改。在部署该模型前，不会在模型外进行任何更改：在部署前，统一数据库没有变化，也不创建或更改远程数据库。

请参见“模型模式”一节第 28 页和“部署模型”一节第 38 页。

### 注意

- ◆ 一个模型只能有一个发布。请参见“发布数据”一节《MobiLink - 客户端管理》。
- ◆ 一个模型只能有一个版本。请参见“脚本版本”一节《MobiLink - 服务器管理》。

## 远程数据库

模型包含远程数据库的模式。该模式可从现有远程数据库或统一数据库中获取。

在以下情况下使用现有远程数据库：

- ◆ 如果您已拥有了远程数据库，尤其是其模式不是统一数据库模式的子集的远程数据库。
- ◆ 如果统一和远程列需要具有不同的类型。例如，如果您需要将统一数据库上的 NCHAR 列映射到 UltraLite 远程数据库上的 CHAR 列。
- ◆ 如果您的远程表需要具有与统一数据库上的表不同的所有者。对于新的 SQL Anywhere 远程数据库，远程表的所有者与相应统一数据库表的所有者相同。如果需要不同的所有者，则应该使用具备所需的表所有权的现有 SQL Anywhere 远程数据库。（UltraLite 数据库没有所有者。）

### 提示：

如果您需要更改现有远程数据库的模式，则在模型之外对数据库进行更改，然后运行 [更新模式向导] 来更新模型。

部署模型时，无论您如何在模型中创建远程模式，您的远程数据库始终有三个选项。远程数据库的部署时选项为：

- ◆ **创建新的远程数据库** 部署可以使用同步模型中的远程模式来创建新的远程数据库。
- ◆ **更新没有用户表的现有远程数据库** 如果您部署到一个空的远程数据库，则在数据库中创建模型中的远程模式。如果您希望使用非缺省的数据库创建选项（如归类），则此选项会派上用场。  
对于 SQL Anywhere 数据库，您可以参见“初始化实用程序 (dbinit)”一节《SQL Anywhere 服务器 - 数据库管理》的注释部分中的选项列表，这些选项无法在数据库创建之后进行设置。  
对于 UltraLite 数据库，数据库属性无法在数据库创建之后进行更改。请参见“为 UltraLite 选择创建时数据库属性”一节《UltraLite - 数据库管理和参考》。
- ◆ **更新具有与模型中的模式相匹配的模式的现有远程数据库** 如果您希望同步一个现有的远程数据库，则此选项会很有用。如果直接部署到现有远程数据库，则不会对任何现有远程数据做出更改。如果尝试直接部署到其模式与模型中的远程模式不匹配的现有远程数据库，则系统会提示您更新模型中的远程模式。

---

对于 SQL Anywhere 远程数据库，表的所有者与原始数据库的所有者相同。（UltraLite 表没有所有者。）

#### 另请参见

- ◆ [“创建模型”一节第 25 页](#)
- ◆ [“部署模型”一节第 38 页](#)

## 更改您的统一数据库

MobiLink 模型令设置同步应用程序更加容易，因为这些模型创建了很多对象（如表、列和触发器），这些对象是同步功能所必需的。

在将这些更改应用到统一数据库之前，您可以确切地确定要进行哪些更改：

- ◆ 在收到 [创建同步模型向导] 或 [部署同步模型向导] 的提示时，您可以不安装 MobiLink 设置，而是检查设置文件（该文件是一个 .sql 文件），然后自行运行该文件以做出更改。
- ◆ 您可以不直接部署到您的统一数据库，而是让 [部署同步模型向导] 来部署对 .sql 文件的更改，您可以检查这些文件然后自行运行它们。

请参见 [“部署模型”一节第 38 页](#)。

## 模型模式

完成 [创建同步模型向导] 或打开现有模型时，模型将以**模型模式**显示。可以使用模型模式进一步自定义模型。如果在 [模型] 模式下工作，则您处于脱机状态，并且所做的更改仅更改模型文件。在部署前，不对统一数据库或远程数据库进行任何更改。

### 管理模式

Sybase Central 的 MobiLink 插件包括两种模式：[模型] 模式和 [管理] 模式。在工具栏中有一个 [模式] 菜单可以在两种模式之间切换。

可以在 [管理] 模式中自定义同步应用程序。然而，如果您先部署模型然后在该模型外部做出更改，则不能执行反向工程将更改回复到模型中。因此，如果您计划重新部署模型，则切勿在 [管理] 模式下更改模型。

有关管理模式的详细信息，请参见文档相关部分中的说明。有关管理模式联机帮助的完整列表，请参见“[MobiLink 插件 \[管理\] 模式帮助](#)”《[SQL Anywhere 10 - 上下文相关帮助](#)》。

## 修改表映射和列映射

表映射指示应同步哪些表、应如何同步表及如何在远程数据库和统一数据库之间映射这些同步数据。

### 仅上载、仅下载和非同步的表或列

缺省情况下，MobiLink 执行完整的双向同步。每个表都可以更改为仅上载或仅下载。您也可以选择不同步表。

在模型中，只能将表指定为仅下载，但不能创建仅下载发布。这是因为一个模型只能有一个发布。

#### ◆ 更改表映射方向

1. 在 [模型] 模式中打开 [映射] 选项卡。
2. 在 [表映射] 窗格中选择一个远程表。
3. 在 [映射方向] 下拉列表中选择以下方向之一：
  - ◆ 双向
  - ◆ 只上载到统一的
  - ◆ 只下载到远程
  - ◆ 未同步

#### 小心

缺省情况下，不同步影子表。切勿尝试对其执行同步操作。

4. 必要时，从 [统一表] 下拉列表中选择要映射的统一表。

#### ◆ 不同步列

1. 在 [模型] 模式中打开 [映射] 选项卡。
2. 在 [表映射] 窗格中选择一个表。  
该表的列信息会显示在下部窗格的 [列映射] 选项卡中。
3. 选择一列。
4. 在 [映射方向] 下拉列表中选择 [未同步]。  
必须同步主键。

#### 更改表映射和列映射

如果模型基于某现有远程数据库，则列映射表示最佳猜测。您应检查这些映射并根据需要自定义它们。

#### ◆ 更改表映射

1. 在 [模型] 模式中打开 [映射] 选项卡。
2. 在 [表映射] 窗格中选择一个表。
3. 更改所映射的统一表：从 [统一表] 下拉列表中，选择另一不同的表。
4. 更改所映射的远程表：
  - ◆ 如果远程数据库基于统一数据库：使用 [创建新远程表] 对话框。请参见“[在远程数据库模式基于统一数据库时创建远程表](#)”一节第 29 页。
  - ◆ 如果远程数据库基于现有远程数据库：从 [远程表] 下拉列表中，选择另一不同的表。
5. 更改表的列映射：选择表，然后打开下部窗格中的 [列映射] 选项卡。

## 修改您的模型要创建的远程数据库

可以按以下操作在模型中修改远程数据库的模式。

#### 在远程数据库模式基于统一数据库时创建远程表

要向模型中的远程数据库模式中添加表，可使用 [创建新远程表] 对话框。这些表被添加到模型中并进行同步映射。要在 [模型] 模式中打开 [创建新远程表] 对话框，可选择 [文件] ► [新建] ► [远程表]。

如果希望向远程数据库中添加表而该表却不在统一数据库中，则可以将该表添加到统一数据库中，运行 [更新模式向导]，然后使用 [创建新远程表] 对话框将该表添加到模型中。要在 [模型] 模式中打开 [更新模式向导]，可从 [文件] 菜单中选择 [更新模式]。

请参见“[在 \[模型\] 模式中更新模式](#)”一节第 37 页。

### 在远程数据库模式基于现有远程数据库时创建远程表

如果您希望为现有远程数据库创建新表，则在模型外修改远程数据库，然后使用 [更新模式向导] 来更新模型中的远程数据库模式。随后需要在 [映射] 选项卡中映射新远程表。请参见：

- ◆ “在 [模型] 模式中更新模式” 一节第 37 页
- ◆ “修改表映射和列映射” 一节第 28 页

### 删除远程表和列

可利用 [模型] 模式删除位于模型中的远程数据库模式中的表和列。右击该行并选择 [删除]，可将某远程表或列标记为要删除。在保存模型时，会从模式中删除该远程表或列。删除远程表或列意味着在您部署到新的远程数据库时不会在该远程数据库中创建该表或该列。

不能删除主键。

不能在 [模型] 模式下从统一数据库中删除表或列。要更改统一模式，可在 [模型] 模式外修改统一数据库然后运行 [更新模式向导]。

### 修改下载类型

下载类型可以是时间戳、快照或自定义。可在 [映射] 选项卡的 [表映射] 窗格中更改下载类型。

- ◆ **基于时间戳的下载** 选择此选项可使用基于时间戳的下载作为缺省选项。仅下载自上次同步后所更改的行。请参见“基于时间戳的下载”一节《MobiLink - 服务器管理》。
- ◆ **快照下载** 选择此选项可使用快照下载作为缺省选项。即使自上次同步后并未对行进行更改，也会下载所有的行。请参见：
  - ◆ “快照同步” 一节《MobiLink - 服务器管理》
  - ◆ “何时使用快照同步” 一节《MobiLink - 服务器管理》
- ◆ **自定义下载逻辑** 如果想要编写自己的 download\_cursor 和 download\_delete\_cursor 脚本而不是生成这些脚本，请选择此选项。请参见：
  - ◆ “编写同步脚本” 《MobiLink - 服务器管理》
  - ◆ “编写 download\_cursor 脚本” 一节《MobiLink - 服务器管理》
  - ◆ “编写 download\_delete\_cursor 脚本” 一节《MobiLink - 服务器管理》

#### ◆ 更改下载类型

1. 在 [模型] 模式中打开 [映射] 选项卡。
2. 在 [表映射] 窗格中选择一个远程表。
3. 在 [下载类型] 下拉列表中，选择 [时间戳]、[快照] 或 [自定义]。
4. 如果选择 [自定义]，则打开 [事件] 选项卡：
  - ◆ 右击该表并选择 [转到事件]。
5. 使用合适的业务逻辑编辑您的 download\_cursor 脚本和 download\_delete\_cursor 脚本。

## 修改处理删除的方式

如果使用快照下载，则会在下载快照之前删除远程数据库中的所有行。如果使用基于时间戳的下载，则可以确定在远程数据库上处理统一数据库中的删除的方式。

如果在将行从统一数据库中删除时也希望将其从远程数据库中删除，则您需要保留该行的记录以删除该行。可以使用影子表或逻辑删除来执行此操作。

### ◆ 更改处理删除的方式

1. 在 [模型] 模式中打开 [映射] 选项卡。
2. 在 [表映射] 窗格中选择一个远程表。
3. 如果希望从统一数据库中下载删除，可在 [下载删除] 下拉列表选中此复选框。如果不希望从统一数据库下载删除，请清除此复选框。
4. 如果选择要下载删除，可打开下部窗格中的 [下载删除] 选项卡。

要记录删除，可以选择使用影子表或逻辑删除。

### 另请参见

- ◆ “处理删除”一节 《MobiLink - 服务器管理》
- ◆ “编写 `download_delete_cursor` 脚本”一节 《MobiLink - 服务器管理》
- ◆ “`download_cursor` 表事件”一节 《MobiLink - 服务器管理》

## 修改下载子集

每个 MobiLink 远程数据库都可以同步统一数据库中的数据子集。可以为每个表自定义下载子集。

下载子集选项为：

- ◆ **用户** 选择此选项可按 MobiLink 用户名对数据进行分区，这样会将不同的数据下载到不同的注册 MobiLink 用户。为了使用此选项，MobiLink 用户名必须在统一数据库上。您可在部署时选择 MobiLink 用户名，以便选择与统一数据库上现有值匹配的名称。如果 MobiLink 用户名在与您进行子集划分的表不同的表中，则必须连接到该表。
- ◆ **远程** 选择此选项可按远程 ID 对数据进行分区，这样会将不同的数据下载到不同的远程数据库。如要使用此选项，则远程 ID 必须在您的统一数据库上。缺省情况下，会将远程 ID 创建为 GUID，但您可以设置与统一数据库上现有值匹配的 ID。如果远程 ID 处于不同于您进行子集划分的表的其它表中，则必须连接到该表。
- ◆ **自定义** 选择此选项以使用确定下载哪些行的 SQL 表达式。每个同步仅下载 SQL 表达式为 true 的那些行。此 SQL 表达式与 `download_cursor` 脚本中所使用的表达式相同。部分是为您生成。

### ◆ 更改下载子集

1. 在 [模型] 模式中打开 [映射] 选项卡。
2. 在 [表映射] 窗格中选择一个远程表。

3. 在 [下载子集] 下拉列表中，选择 [无]、[用户]、[Remote] 或 [自定义]。
4. 如果选择 [用户]、[Remote] 或 [自定义]，可打开下部窗格中的 [下载子集] 选项卡。
5. 如果选择 [用户] 或 [Remote]，则可以利用 [下载子集] 选项卡来标识包含 MobiLink 用户名或远程 ID 的统一表中的列，或者输入一个表的连接以获取 MobiLink 用户名或远程 ID。
6. 如果选择 [自定义]，则 [下载子集] 选项卡有两个文本框，您可以向其中添加信息以构造 `download_cursor` 脚本。不必编写完整的 `download_cursor`。只需添加额外信息以标识下载子集上的连接和其它限制即可。
  - ◆ 如果您的 `download_cursor` 脚本需要到其它表的连接，可在第一个文本框（[要添加到下载游标的 FROM 子句中的表]）中输入表名。如果连接需要多个表，则用逗号分隔它们。
  - ◆ 在第二个文本框（[要在下载游标的 WHERE 子句中使用的 SQL 表达式]）中，输入指定连接和下载子集的 WHERE 条件。

### 另请参见

- ◆ “MobiLink 用户简介”一节 《MobiLink - 客户端管理》
- ◆ “远程 ID”一节 《MobiLink - 客户端管理》
- ◆ “在远程数据库之间对行进行分区”一节 《MobiLink - 服务器管理》
- ◆ “在脚本中使用远程 ID 和 MobiLink 用户名”一节 《MobiLink - 客户端管理》
- ◆ “在脚本中使用上次下载时间”一节 《MobiLink - 服务器管理》
- ◆ “编写 `download_cursor` 脚本”一节 《MobiLink - 服务器管理》

### 示例（用户）

例如，CustDB 中的 ULOrder 表可在用户之间共享。缺省情况下，会将订单指派给创建它们的雇员。但有时另一位雇员需要查看其他雇员所创建的订单。例如，经理可能需要查看部门内雇员创建的所有订单。CustDB 数据库可通过 ULEmpCust 表提供这些订单。这样您就可以将客户指派给雇员。他们下载该雇员客户关系的所有订单。

若想知道这是如何完成的，可先查看 ULOrder 的 `download_cursor` 脚本，但不要下载子集。选择 [映射] 选项卡中的 ULEmpCust 表。选择基于时间戳的下载但不选择任何下载子集。右击该表并选择 [转到事件]。该表的 `download_cursor` 脚本如下所示：

```
SELECT "DBA"."ULOrder"."order_id",
       "DBA"."ULOrder"."cust_id",
       "DBA"."ULOrder"."prod_id",
       "DBA"."ULOrder"."emp_id",
       "DBA"."ULOrder"."disc",
       "DBA"."ULOrder"."quant",
       "DBA"."ULOrder"."notes",
       "DBA"."ULOrder"."status"
FROM "DBA"."ULOrder"
WHERE "DBA"."ULOrder"."last_modified" >= {ml s.last_table_download}
```

现在，回到 [映射] 选项卡。将 ULOrder 的 [下载子集] 列更改为 [用户]。打开下部窗格中的 [下载子集] 选项卡。选择 [在已连接关系表中使用列]。对要连接的表，选择 ULEmpCust。对要匹配的列，选择 `emp_id`。连接条件应为 `emp_id = emp_id`。

在顶部窗格中右击该表并选择 [转到事件]。现在，该表的 `download_cursor` 脚本如下所示：

```

SELECT "DBA"."ULOrder"."order_id",
       "DBA"."ULOrder"."cust_id",
       "DBA"."ULOrder"."prod_id",
       "DBA"."ULOrder"."emp_id",
       "DBA"."ULOrder"."disc",
       "DBA"."ULOrder"."quant",
       "DBA"."ULOrder"."notes",
       "DBA"."ULOrder"."status"
FROM "DBA"."ULOrder", "DBA"."ULEmpCust"
WHERE "DBA"."ULOrder"."last_modified" >= {ml s.last_table_download}
AND "DBA"."ULOrder"."emp_id" = "DBA"."ULEmpCust"."emp_id"
AND "DBA"."ULEmpCust"."emp_id" = {ml s.username}

```

### 示例（自定义）

例如，假设您要按 MobiLink 用户对 Customer 表的下载进行子集划分，并且仅要下载 active=1 的行。而 MobiLink 用户名不在您进行子集划分的表中，因此您需要创建与名为 SalesRep 的表的连接，该表中包含这些用户名。

在 [映射] 选项卡中，为 Customer 表选择 [下载类型] 为 [时间戳]，[下载子集] 为 [自定义]。打开下部窗格中的 [下载子集] 选项卡。在第一个文本框（[要添加到下载游标的 FROM 子句中的表]）中，键入：

```
SalesRep
```

在第二个文本框（[要在下载游标的 WHERE 子句中使用的 SQL 表达式]）中，键入：

```

SalesRep.ml_username = {ml s.username}
AND Customer.active = 1
AND Customer.cust_id = SalesRep.cust_id

```

在顶部窗格中右击该表并选择 [转到事件]。现在，该表的 download\_cursor 脚本如下所示：

```

SELECT "DBA"."Customer"."cust_id",
       "DBA"."Customer"."cust_name"
FROM "DBA"."Customer", SalesRep
WHERE "DBA"."Customer"."last_modified" >= {ml s.last_table_download}
AND SalesRep.ml_username = {ml s.username}
AND Customer.active = 1
AND Customer.cust_id = SalesRep.cust_id

```

WHERE 子句的最后一行创建了 Customer 到 SalesRep 的键连接。

## 修改冲突检测和冲突解决

如果在远程数据库和统一数据库中均更新某行，则下次同步数据库时会发生冲突。

可使用以下选项来检测冲突：

- ◆ **无冲突检测** 如果不希望有任何冲突检测，请选择此选项。应用已上载的更新而不检查冲突。这样就不必从统一数据库读取当前的行值，因此同步更新的速度可能会更快。
- ◆ **基于行的冲突检测** 如果自上次同步后行已由远程和统一数据库更新，则会检测到冲突。

此选项定义 upload\_fetch 和 upload\_update 脚本。请参见“使用 upload\_fetch 脚本检测冲突”一节《MobiLink - 服务器管理》。

- ◆ **基于列的冲突检测** 如果远程数据库和统一数据库中行的相同列都已更新，则会检测到冲突。此选项定义 `upload_fetch_column_conflict` 脚本。请参见“使用 `upload_fetch` 脚本检测冲突”一节《MobiLink - 服务器管理》。
- 如果表中包含 BLOB 且选择 [基于列的冲突检测]，则使用 [基于行的冲突检测]。
- 可使用以下选项来解决冲突：
  - ◆ **统一** 先入优先：拒绝发生冲突的上载更新。
  - ◆ **远程** 后入优先：始终应用上载更新。
  - 仅在使用基于列的冲突检测时，才使用此选项。否则，如果您选择无冲突检测，将会获得同样的结果和更佳的性能。
  - ◆ **Timestamp** 最新的更新优先。要使用此选项，您必须为表创建和维护时间戳列。此时间戳列应记录上次更改行的时间。该列在统一数据库和远程数据库中都应存在。远程数据库和统一数据库必须使用相同的时区（首选 UTC），且其时钟必须同步，才可正常工作。
  - ◆ **自定义** 编写自己的 `resolve_conflict` 脚本。向导完成后，可在 [事件] 选项卡中完成此工作。请参见“使用 `resolve_conflict` 脚本解决冲突”一节《MobiLink - 服务器管理》。

#### ◆ 自定义冲突检测和冲突解决

1. 在 [模型] 模式中打开 [映射] 选项卡。
2. 在 [表映射] 窗格中选择一个远程表。
3. 在 [冲突检测] 下拉列表中，选择 [无]、[基于行] 或 [基于列]。如果选择 [无]，则操作到此结束。
4. 如果选择 [基于行] 或 [基于列]，则还要从 [冲突解决] 下拉列表中选择 [统一]、[Remote]、[时间戳] 或 [自定义]。
5. 如果选择 [时间戳] 冲突解决，可打开下部窗格中的 [冲突解决] 选项卡然后输入要使用的时间戳列名称。
6. 如果选择 [自定义] 冲突解决，可打开 [事件] 选项卡然后为此表编写 `resolve_conflict` 脚本。

#### 另请参见

- ◆ “冲突处理”一节《MobiLink - 服务器管理》

## 在模型中修改脚本

在 MobiLink [模型] 模式中打开 [事件] 选项卡以执行以下操作：

- ◆ 查看和修改由 [创建同步模型向导] 生成的脚本。
- ◆ 创建新脚本。

可从 [事件] 选项卡顶部获知所选脚本所属的组。为了您的方便，将一个表的所有脚本归组到一起。还可从 [事件] 选项卡顶部获知所选脚本的名称及该脚本是否由 [创建同步模型向导] 生成、是否用户定义或者是否替换已生成的脚本。还可获知编写同步逻辑使用的语言是 SQL、.NET 还是 Java。

添加或更改脚本时，脚本会完全在您的控制之下；当在 [模型] 模式中更改某相关设置时，该脚本也不会自动更改。例如，如果您更改了模型的 `download_delete_cursor` 脚本，然后在 [模型] 模式中取消选择 [下载删除]，则您的自定义 `download_delete_cursor` 脚本不会受到影响。

可以使用 [文件] 菜单中的选项来恢复已经更改的缺省脚本，或删除已添加的新脚本。选择要恢复的脚本然后选择 [文件] 以查看您的选项。

要查找某特定表的脚本，可打开 [映射] 选项卡，选择行，然后选择 [文件] ► [转到事件]。[事件] 选项卡随即在适当的表处打开。

## 在模型模式中向外部服务器验证

要向外部 POP3、IMAP 或 LDAP 服务器验证，可在模型模式中打开 [验证] 选项卡，然后选择 [为此同步模型启用自定义验证]。

必须输入有关主机和端口的信息，如果是 LDAP 服务器，则需输入 LDAP 服务器的 URL。

有关这些字段的详细信息，请参见“外部验证程序属性”一节《MobiLink - 客户端管理》。

## 在模型模式中设置服务器启动的同步

服务器启动的同步允许您在统一数据库中发生某些更改时在客户端启动同步。[模型] 模式提供了设置服务器启动的同步的方法。此方法提供了便于设置和运行的有限版本的服务器启动同步。

### [通知] 选项卡

#### ◆ 设置服务器启动的同步（Sybase Central 模型模式和部署同步模型向导）

1. 使用 [创建同步模型向导] 创建 MobiLink 模型。
2. 在 [模型] 模式中已打开模型时，在该模型的顶部打开 [通知] 选项卡。
3. 选择 [启用服务器启动的同步]。
4. 选择一个用于通知的统一数据库表。

对此表中进行的更改会导致向远程数据库发送通知，并促使远程数据库开始同步数据。该通知触发同步。

仅可以为此目的选择表，对此，您已经定义了基于时间戳的下载游标（缺省值）。该通知基于下载游标的内容。

请参见“编写 `download_cursor` 脚本”一节《MobiLink - 服务器管理》。

5. 选择轮询间隔。此时间为两次轮询之间的时间。可以选择预定义的轮询间隔时间，也可以输入间隔时间。缺省值是 30 秒。

如果通告程序丢失数据库连接，它将在数据库重新可用后的第一个轮询间隔自动恢复。

6. 或者，更改通告程序数据库连接的隔离级别。缺省值为 [读取已提交的]。

请注意设置隔离级别的后果。较高的级别会增加争用，并可能导致性能降低。隔离级别 0 允许读取未提交的数据—最终可能被回退的数据。

7. 还可以更改通过其发送通知的网关。缺省为 default\_device\_tracker 网关。请参见“[网关和运营公司](#)”一节《[MobiLink - 服务器启动的同步](#)》。

#### ◆ 使用服务器启动的同步部署模型

1. 部署模型：

- a. 从 [文件] 菜单中选择 [部署]。
- b. 请按照 [部署同步模型向导] 中的说明进行操作。  
请参见“[部署模型](#)”一节第 38 页。
- c. 在 [服务器启动的同步监听器] 页面上，为监听器配置选项。

2. 完成部署模型。有关所创建文件的信息，请参见“[同步已部署的模型](#)”一节第 39 页。

3. 要使用服务器启动的同步，必须完成以下操作：

- a. 启动 MobiLink 服务器。
- b. 执行第一次同步（如果尚未执行同步）。
- c. 启动监听器。

4. 浏览到第一次启动 [创建同步模型向导] 时选择的目录。该文件夹保存扩展名为 *.mlsm* 的模型。该目录也保存以下子目录：

- ◆ *\mlsrv*
- ◆ *\remote*
- ◆ *\consolidated*

#### 关于服务器启动的同步（不在 [模型] 模式中）

在 [模型] 模式版本的服务器启动同步中，MobiLink 服务器使用表的 download\_cursor 脚本来确定何时启动同步。实现方法是使用 download\_cursor 脚本为通告程序生成 request\_cursor。如果使用此版本的服务器启动的同步，则不能自定义您的 request\_cursor。

请参见“[编写 download\\_cursor 脚本](#)”一节《[MobiLink - 服务器管理](#)》和“[request\\_cursor 属性](#)”一节《[MobiLink - 服务器启动的同步](#)》。

[模型] 模式还设置一个缺省的设备跟踪器网关来发送通知。您可以自定义网关。请参见“[网关和运营公司](#)”一节《[MobiLink - 服务器启动的同步](#)》。

虽然 Sybase Central 的 [模型] 模式提供了一个简化版的服务器启动的同步，但您也可以设置一个完整版的服务器启动的同步。

有关服务器启动同步的完整实现的说明，请参见 [MobiLink - 服务器启动的同步](#) 《MobiLink - 服务器启动的同步》。

有关不利用模型模式设置服务器启动同步需要进行哪些操作的概述，请参见“[服务器启动的同步快速入门](#)”一节 《MobiLink - 服务器启动的同步》。

## 在 [模型] 模式中更新模式

可利用 [更新模式向导] 在模型中更新统一数据库和远程数据库模式。

在您部署完模型并执行了以下操作后 [更新模式向导] 才发挥其最大效用：

- ◆ 对需要包括在该模型中的远程数据库模式做了更改。
- ◆ 对需要包括在该模型中的统一数据库模式做了更改。

例如，在重新部署为一个或多个表创建了基于时间戳下载的模式之前，需要运行 [更新模式]。由于先前的部署通过添加时间戳列或影子表而更改了统一数据库的模式，因此需要更新该模式。

与向模型添加远程表的 [创建新远程表] 对话框不同，[更新模式向导] 不会映射表。需要使用 [映射] 选项卡创建表映射。请参见“[修改您的模型要创建的远程数据库](#)”一节第 29 页。

### ◆ 更新模式

1. 在 [模型] 模式中，从 [文件] 菜单中选择 [更新模式]。

即会出现 [更新模式向导]。

2. 选择您要更新的模式。

[更新模式向导] 会将模型中的模式同数据库的模式加以比较。

- ◆ **统一数据库模式** 更新了模型中的统一模式。模型中的远程模式未作更改。
- ◆ **远程数据库模式** 更新了模型中的远程模式。模型中的统一模式未作更改。
- ◆ **统一和远程数据库模式。** 模型中的统一模式和远程模式均得到更新，以与现有数据库的模式相匹配。

3. 按照向导中的说明进行操作。每页都有联机帮助。

4. 单击 [完成]。

单击 [完成] 后，关闭与统一数据库和远程数据的所有连接。现在您即以脱机方式工作。在部署该模型前，不会在模型外进行任何更改：在部署前，统一数据库没有变化，也不创建或更改远程数据库。

5. 在 [映射] 选项卡中映射新远程表。请参见“[修改表映射和列映射](#)”一节第 28 页。

## 部署模型

准备好尝试所建模型后，可以使用 [部署同步模型向导] 对其进行部署。可以部署许多内容：

- ◆ 对统一数据库进行的更改。
- ◆ SQL Anywhere 或 UltraLite 远程数据库（可选择创建数据库、将表添加到现有空数据库中，或使用已包含您的远程表的现有数据库）。
- ◆ 用于部署模型的批处理文件。
- ◆ 用于运行 MobiLink 服务器和 MobiLink 客户端的批处理文件。
- ◆ 服务器启动的同步配置。

部署模型时会保存模型文件。

### 部署到统一数据库

[部署] 向导提供了两个用于部署统一数据库的选项：

- ◆ 通过填充 MobiLink 系统表和创建所需的所有影子表、列、触发器和存储过程，将模型直接应用到统一数据库。还可以选择创建能够运行 MobiLink 应用程序的批处理文件。
- ◆ 创建包含所有相同更改的 SQL 文件。可以随时检查、变更和运行此文件。其效果与直接应用更改相同。

#### 注意

如果您的部署创建了影子表，则必须以基表（将为这些基表创建影子表）所有者的身份或管理员身份连接统一数据库。

### 部署远程数据库

可以选择使用现有的远程数据库或利用向导创建一个数据库。向导可以直接创建远程数据库，或者您可以利用向导创建一个 SQL 文件，然后运行该文件来创建远程数据库。

利用缺省数据库创建选项，向导使用您在模型中指定的数据库所有者创建远程数据库（SQL Anywhere 或 UltraLite）。也可以使用您自己的自定义设置在 [创建同步模型向导] 之外创建远程数据库并使用该向导添加所需的远程表，或者部署到已经含有远程表的现有远程数据库。

### 部署批处理文件以运行同步工具

向导可以创建以下批处理文件：

- ◆ 使用指定的选项运行 MobiLink 服务器的批处理文件。
- ◆ 对于 SQL Anywhere 远程数据库，批处理文件使用指定的选项运行 dbmlsync。
- ◆ 对于 UltraLite 远程数据库，批处理文件使用指定的选项运行 ulsync。Ulsync 用于测试同步，因此没有可用的 UltraLite 应用程序时，它可以帮助您开始工作。
- ◆ 如果正在设置服务器启动的同步，[部署同步模型向导] 还可以创建可运行 [通告程序] 和 [监视器] 的批处理文件。

### ◆ 部署模型

1. 在 [模型] 模式中，选择 [文件] ► [部署]。  
随即出现 [部署同步模型向导]。
2. 按照向导中的步骤进行操作。每页都有联机帮助。
3. 完成后，所选更改即部署完毕。如果有同名现有文件，将覆盖这些文件。
4. 要同步应用程序，请参见“[同步已部署的模型](#)”一节第 39 页。

## 重新部署模型

部署模型后，可以对其进行更改。通过在 [模型] 模式中进行更改然后再重新部署即可实现此操作。还可以在 Sybase Central [管理] 模式中变更已部署的应用程序，或者使用系统过程或其它方法直接更改数据库。然而，在 [模型] 模式之外变更已部署的模型时，不能执行反向工程将更改回复到模型。重新部署模型时，将覆盖在模型外部对您的同步应用程序做出的更改。

无论您对远程数据库或统一数据库的模式做出哪些更改，在重新部署前您都需要在模型中更新该模式。部署经常会导致模式更改，因此即使您未做任何其它更改可能也需要更新该模式。例如，如果部署一个模型，该模型在统一数据库的每个同步表中都添加一个时间戳列（此为创建模型时的缺省行为），则在重新部署之前需要更新模型中的统一模式。同样，如果向统一数据库添加了一个表然后希望重新部署，则需要更新模型中的统一模式然后创建新的远程表。

要运行 [更新模式向导]，请选择 [文件] ► [更新模式]。

请参见“[在 \[模型\] 模式中更新模式](#)”一节第 37 页。

## 同步已部署的模型

部署模型时，可以选择在 [创建同步模型向导] 的第一个屏幕上所选择的位置处创建目录和文件。将根据当时所选择的模型名称为这些文件和目录命名。

假定您将模型命名为 **MyModel** 并保存在 *c:\SyncModels* 中。根据所选择的部署选项，可以有以下文件：

目录（基于示例的名称和位置）	说明和内容（基于示例的名称）
<i>c:\SyncModels</i>	包含模型文件，保存为 <i>MyModel.mlsm</i> 。
<i>c:\SyncModels\MyModel</i>	包含保存部署文件的文件夹。
<i>c:\SyncModels\MyModel\consolidated</i>	包含用于统一数据库的部署文件： <ul style="list-style-type: none"> <li>◆ <i>MyModel_consolidated.sql</i> - 用于设置统一数据库的 SQL 文件。</li> <li>◆ <i>MyModel_consolidated.bat</i> - 用于运行 SQL 文件的批处理文件。</li> </ul>

目录（基于示例的名称和位置）	说明和内容（基于示例的名称）
<code>c:\SyncModels\MyModel\mlsrv</code>	包含用于 MobiLink 服务器的部署文件： <ul style="list-style-type: none"> <li>◆ <i>MyModel_mlsrv.bat</i> - 用于运行 MobiLink 服务器的批处理文件。如果已经设置服务器启动的同步，则运行该文件也将启动通告程序。</li> </ul>
<code>c:\SyncModels\MyModel\remote</code>	包含用于远程数据库的部署文件： <ul style="list-style-type: none"> <li>◆ <i>dblsn.txt</i> - 如果设置服务器启动的同步，它将是包含 [监听器] 选项设置的文本文件。该文件由 <i>MyModel_dblsn.bat</i> 使用。</li> <li>◆ <i>MyModel_dblsn.bat</i> - 如果设置服务器启动的同步，该文件为用于运行监听器的批处理文件。</li> <li>◆ <i>MyModel_dbmlsync.bat</i> - 如果部署了一个 SQL Anywhere 远程数据库，则此文件是使用 dbmlsync 同步 SQL Anywhere 数据库的批处理文件。</li> <li>◆ <i>MyModel_remote.bat</i> - 用于运行 <i>MyModel_remote.sql</i> 的批处理文件。</li> <li>◆ <i>MyModel_remote.db</i> - 如果选择创建一个新的 SQL Anywhere 远程数据库，则此文件即是该数据库。</li> <li>◆ <i>MyModel_remote.sql</i> - 用于设置新 SQL Anywhere 远程数据库的 SQL 文件。</li> <li>◆ <i>MyModel_remote.udb</i> - 如果选择创建一个新的 UltraLite 远程数据库，则此文件即是该数据库。</li> <li>◆ <i>MyModel_ulsync.bat</i> - 如果部署了一个 UltraLite 数据库，则为用于使用 ulsync 实用程序测试与 UltraLite 远程数据库的同步的批处理文件。</li> </ul>

### 运行批处理文件

必须从命令行运行由 [部署同步模型向导] 创建的批处理文件，并且必须将连接信息包括在内。在运行这些批处理文件之前，可能需要创建 ODBC 数据源。

请参见“使用 ODBC 数据源”一节《SQL Anywhere 服务器 - 数据库管理》。

#### ◆ 使用批处理文件同步模型

1. 如果尚未在统一数据库上运行 MobiLink 设置脚本，则在部署之前运行这些脚本。

请参见“建立统一数据库”一节《MobiLink - 服务器管理》。

2. 在运行 [部署同步模型向导] 时，如果您选择创建一个以后运行的文件（在 [统一数据库部署目标] 页面上），则必须运行位于您模型的 *consolidated* 子文件夹中的批处理文件。此文件创建您已选择在统一数据库中创建的所有对象，其中包括同步脚本、影子表和触发器。该文件也可在统一数据库中注册 MobiLink 用户。

要运行此文件，可浏览到 *consolidated* 目录，然后运行以 *\_consolidated.bat* 结尾的文件。必须将连接信息包括在内。例如，运行：

```
MyModel_consolidated.bat "dsn=MY_ODBC_DATASOURCE"
```

3. 在运行 [部署同步模型向导] 时，如果您选择创建一个以后运行的文件（在 [新 SQL Anywhere 远程数据库] 页面或 [新 UltraLite 远程数据库] 页面上），则必须运行位于 *remote* 目录中的批处理文件。此文件创建所有您已选择要在远程数据库中创建的对象，其中包括表、发布、预订和 MobiLink 用户。

要运行此文件，可浏览到 *remote* 目录，然后运行以 *\_remote.bat* 结尾的文件。例如，运行：

```
MyModel_remote.bat
```

4. 通过运行 *mlsrv\MyModel\_mlsrv.bat* 启动 MobiLink 服务器。如果设置服务器启动的同步，则运行该文件也将启动通告程序。必须将连接信息包括在内。例如，运行：

```
MyModel_mlsrv.bat "dsn=MY_ODBC_DATASOURCE"
```

5. 执行同步。

对于 SQL Anywhere 远程数据库：

- ◆ 将 REMOTE DBA 权限授予非 DBA 用户（建议）。例如，使用 Interactive SQL 执行以下语句：

```
GRANT REMOTE DBA  
TO userid, IDENTIFIED BY password
```

以具有 REMOTE DBA 权限的用户身份进行连接。

- ◆ 启动位于 *remote* 目录中的远程数据库。例如，运行：

```
dbeng10 MyModel_remote.db
```

- ◆ 启动 dbmlsync、SQL Anywhere MobiLink 客户端。运行 *remote* 目录中以 *\_dbmlsync.bat* 结尾的文件。必须将连接信息包括在内。例如，运行：

```
MyModel_dbmlsync.bat "uid=dba;pwd=sql;eng=MyModel_remote"
```

对于 UltraLite 远程数据库：

- ◆ 要测试同步，可运行 *remote* 目录中以 *\_ulsync.bat* 结尾的文件。

- ◆ 或者，运行 UltraLite 应用程序。

6. 如果设置服务器启动的同步，则需要执行第一次同步，然后启动监听器。要创建远程 ID 文件，必须进行第一次同步。要启动监听器，可运行 *remote* 目录中以 *\_dblsn.bat* 结尾的文件。例如，运行：

```
MyModel_dblsn.bat
```

### 另请参见

- ◆ “GRANT REMOTE DBA 语句 [MobiLink] [SQL Remote]” 一节 《SQL Anywhere 服务器 - SQL 参考》
- ◆ “dbmlsync 权限” 一节 《MobiLink - 客户端管理》

---

## 第 II 部分 MobiLink 教程

本部分提供的教程介绍如何设置和使用 MobiLink 技术。其涵盖范围包括面向新用户的介绍性教程到如何使用高级功能的说明。



---

## 第 3 章

# 探讨 MobiLink 的 CustDB 示例

## 目录

MobiLink CustDB 教程简介 .....	46
CustDB 安装 .....	48
CustDB 数据库中的表 .....	54
CustDB 示例中的用户 .....	57
同步 CustDB .....	58
维护客户和订单的主键池 .....	62
清除 .....	64
进一步阅读 .....	65

## MobiLink CustDB 教程简介

CustDB 是一个销售状态应用程序。CustDB 示例是适合于 MobiLink 开发人员的宝贵资源。它提供了如何实现开发 MobiLink 应用程序所需的多种技术的若干示例。

此应用程序旨在说明几种常用的同步技术。为了从本章学到更多知识，请在阅读本章的同时学习该示例应用程序。

对每种支持的操作系统和数据库类型都提供了一种版本的 CustDB。

有关 CustDB 的位置和设置说明的信息，请参见“[建立 CustDB 统一数据库](#)”一节第 48 页。

### 方案

CustDB 方案如下。

统一数据库位于总部。以下数据存储在同一数据库中：

- ◆ 保存同步元数据的 MobiLink 系统表，其中包括实现同步逻辑的同步脚本。
- ◆ 包括所有客户、产品和订单信息在内的 CustDB 数据，这些数据存储在建表的行中。

有两种类型的远程数据库：移动经理和移动销售代表。

每个移动销售代表的数据库中包含所有的产品，但仅包含分配给该销售代表的订单；而移动经理的数据库中包含所有的产品和订单。

### 同步设计

CustDB 示例应用程序中的同步设计将使用以下功能：

- ◆ **完整的表下载** ULProduct 表中的所有行和列都完全与远程数据库共享。
- ◆ **列子集** ULCustomer 表中所有行（但不是所有列）都与远程数据库共享。
- ◆ **行子集** 不同的远程用户可从 ULOrder 表中获得不同的行集合。

有关行子集的详细信息，请参见“[在远程数据库之间对行进行分区](#)”一节《[MobiLink - 服务器管理](#)》。

- ◆ **基于时间戳的同步** 这是一种用于识别自上次设备同步以来对统一数据库所做更改的方法。ULCustomer 和 ULOrder 表将采用基于时间戳的方法进行同步。

请参见“[基于时间戳的下载](#)”一节《[MobiLink - 服务器管理](#)》。

- ◆ **快照同步** 这是一种简单的同步方法，可在每次同步中下载所有行。ULProduct 表使用这种方法进行同步。

请参见“[快照同步](#)”一节《[MobiLink - 服务器管理](#)》。

- ◆ **通过主键池保持主键唯一** 确保主键值在整个 MobiLink 安装中的唯一性至关重要。在此应用程序中采用的主键池方法是一种确保主键唯一性的方法。

请参见“[使用主键池](#)”一节《[MobiLink - 服务器管理](#)》。

有关确保主键唯一性的其它方法，请参见“维护唯一主键”一节《MobiLink - 服务器管理》。

有关 CustDB 表的实体关系图，请参见“关于 CustDB 示例数据库”一节《SQL Anywhere 10 - 简介》。

#### **进一步阅读**

- ◆ “探讨 UltraLite 的 CustDB 示例” 《UltraLite - 数据库管理和参考》

## CustDB 安装

本节将介绍组成 CustDB 示例应用程序及数据库的代码片段。其中包括：

- ◆ 示例 SQL 脚本，位于 *samples-dir\MobiLink\CustDB*。
- ◆ 应用程序代码，位于 *samples-dir\UltraLite\CustDB*。
- ◆ 特定于平台的用户界面代码，位于 *samples-dir\UltraLite\CustDB* 目录下分别以每个操作系统命名的子目录中。

**注意**

有关 *samples-dir* 的详细信息，请参见“示例目录”一节《SQL Anywhere 服务器 - 数据库管理》。

### 建立 CustDB 统一数据库

CustDB 统一数据库可以是 SQL Anywhere、Sybase Adaptive Server Enterprise、Microsoft SQL Server、Oracle 或 IBM DB2。

#### SQL Anywhere CustDB

SQL Anywhere CustDB 统一数据库在 *samples-dir\UltraLite\CustDB\custdb.db* 中提供。名为 SQL Anywhere 10 CustDB 的 DSN 会随您的安装提供。

可以使用文件 *samples-dir\UltraLite\CustDB\newdb.bat* 重建此数据库。

如果您要研究创建 CustDB 示例的方法，可查看文件 *samples-dir\MobiLink\CustDB\syncsa.sql*。

#### 其它 RDBMS 的 CustDB

在 *samples-dir\MobiLink\CustDB* 目录中提供的以下 SQL 脚本用来构建 CustDB 统一数据库，作为上述任意一种受支持的 RDBMS：

RDBMS	Custdb 安装脚本
Adaptive Server Enterprise	<i>custase.sql</i>
Microsoft SQL Server	<i>custmss.sql</i>
Oracle	<i>custora.sql</i>
IBM DB2	<i>custdb2.sql</i>

以下过程为每个支持的 RDBMS 创建 CustDB 统一数据库。

有关准备数据库以用作统一数据库的详细信息，请参见“建立统一数据库”一节《MobiLink - 服务器管理》。

**◆ 建立统一数据库（Adaptive Server Enterprise、Oracle 或 SQL Server）**

1. 在 RDBMS 中创建数据库。
2. 运行位于 SQL Anywhere 10 安装目录的 *MobiLink\setup* 子目录中的以下 SQL 脚本之一，即可添加 MobiLink 系统表：
  - ◆ 若为 Adaptive Server Enterprise 统一数据库，运行 *syncase.sql*。
  - ◆ 若为 Oracle 统一数据库，运行 *syncora.sql*。
  - ◆ 若为 SQL Server 统一数据库，运行 *syncmss.sql*。
3. 运行位于 SQL Anywhere 10 安装目录的 *Samples\MobiLink\CustDB* 子目录中的以下 SQL 脚本之一，即可在 CustDB 数据库中添加示例用户表：
  - ◆ 若为 Adaptive Server Enterprise 统一数据库，运行 *custase.sql*。
  - ◆ 若为 Oracle 统一数据库，运行 *custora.sql*。
  - ◆ 若为 SQL Server 统一数据库，运行 *custmss.sql*。
4. 创建一个名为 CustDB 的 ODBC 数据源，该数据源引用客户端计算机上的数据库。
  - ◆ 选择 [开始] ► [程序] ► [SQL Anywhere 10] ► [SQL Anywhere] ► [ODBC 管理器]。
  - ◆ 单击 [添加]。
  - ◆ 从列表中选择相应的驱动程序。
    - 单击 [完成]。
  - ◆ 将 ODBC 数据源命名为 CustDB。
  - ◆ 单击 [登录] 选项卡。输入数据库的用户 ID 和口令。

**◆ 建立统一数据库 (IBM DB2)**

1. 在 DB2 服务器上创建 DB2 数据库。对于本教程，称其为 CustDB。
2. 确保缺省表空间（通常称为 USERSPACE1）使用 8 KB 页。

如果缺省表空间不使用 8 KB 页，请完成以下步骤：

  - ◆ 验证至少有一个缓冲池有 8 KB 页。否则，创建一个具有 8 KB 页的缓冲池。
  - ◆ 创建一个具有 8 KB 页的新表空间和临时表空间。

有关详细信息，请参见 DB2 文档。
3. 使用文件 *MobiLink\setup\syncdb2long.sql* 将 MobiLink 系统表添加到 DB2 统一数据库中：

- ◆ 在文件 *syncdb2long.sql* 的顶部更改连接命令。用 DB2 数据库的名称（或其别名）替换 *DB2Database*。在此示例中，该数据库称为 *CustDB*。还可以按以下方式添加 DB2 用户名和口令：

```
connect to CustDB user userid using password ~
```

- ◆ 在服务器或客户端计算机上，打开一个 DB2 命令窗口。键入以下命令运行 *syncdb2long.sql*：

```
db2 -c -ec -td~ +s -v -f syncdb2long.sql
```

4. 要让 DB2 使用在 *syncdb2long.sql* 中定义的存储过程，您必须将位于 SQL Anywhere 安装目录的 *MobiLink\setup* 子目录中的 *syncdb2long\_version* Java 和类文件复制到 DB2 安装目录的 *FUNCTION* 子目录中。
5. 将位于 SQL Anywhere 安装目录的 *Samples\MobiLink\CustDB* 子目录中的 *custdb2.class* 复制到 DB2 服务器计算机的 *SQLLIB\FUNCTION* 目录中。此类包含用于 *CustDB* 示例的过程。
6. 将数据表添加到 *CustDB* 数据库中：
  - ◆ 如有必要，在 *custdb2.sql* 中更改连接命令。例如，可以按下面的方法添加用户名和口令。用您的用户名和口令替换 *userid* 和 *password*。

```
connect to CustDB user userid using password
```

- ◆ 在服务器或客户机上，打开一个 DB2 命令窗口。
- ◆ 键入以下命令运行 *custdb2.sql*：

```
db2 -c -ec -td~ +s -v -f custdb2.sql
```

- ◆ 处理完毕后，输入如下命令关闭命令窗口：

```
exit
```

7. 创建一个名为 *CustDB* 的 ODBC 数据源，该数据源将引用 DB2 客户端计算机上的 DB2 数据库。

- ◆ 启动 ODBC 管理器：

在 [开始] 菜单中，选择 [程序] ► [SQL Anywhere 10] ► [SQL Anywhere] ► [ODBC 管理器]。

随即出现 [ODBC 数据源管理器]。

- ◆ 在 [用户 DSN] 选项卡上，单击 [添加]。

出现 [创建新数据源] 对话框。

- ◆ 为 DB2 数据库选择 ODBC 驱动程序。例如，选择 IBM DB2 UDB ODBC 驱动程序。单击 [完成]。

有关如何配置 DB2 ODBC 驱动程序的信息，请参见：

- ◆ DB2 文档
  - ◆ [http://www.iAnywhere.com/developer/technotes/odbc\\_mobilink.html](http://www.iAnywhere.com/developer/technotes/odbc_mobilink.html)
8. 按以下方式在 DB2 客户端计算机上运行 *custdb2setuplong* Java 应用程序。该应用程序会重置 DB2 数据库中 CustDB 示例。初始设置完成后，可随时键入同一命令行来运行此应用程序，以重置 DB2 CustDB 数据库。

- ◆ 如果使用 CustDB 之外的名称来作为数据源，则必须修改 *custdb2setuplong.java* 中的连接代码并按以下方式重新编译它。如果系统变量 *%db2tempdir%* 指定的路径中包含空格，则必须用引号将路径括起来。

```
javac -g -classpath %db2tempdir%\java\jdk\lib\classes.zip;
%db2tempdir%\java\db2java.zip;
%db2tempdir%\java\runtime.zip custdb2setuplong.java
```

- ◆ 键入以下命令行，其中 *userid* 和 *password* 是用于连接到 CustDB ODBC 数据源的用户名和口令。

```
java custdb2setuplong userid password
```

### 另请参见

- ◆ “IBM DB2 UDB 统一数据库”一节 《MobiLink - 服务器管理》

## 建立 UltraLite 远程数据库

以下过程用于为 CustDB 创建远程数据库。CustDB 远程数据库必须为 UltraLite 数据库。

远程数据库的应用程序逻辑位于 *samples-dir\UltraLite\CustDB*。它包括以下文件：

- ◆ **嵌入式 SQL 逻辑** 文件 *custdb.sqc* 中包含在 UltraLite 数据库中查询和修改信息时所需的 SQL 语句，以及启动与统一数据库的同步时所需的调用。
- ◆ **C++ API 逻辑** 文件 *custdbcomp.cpp* 包含 C++ API 逻辑。
- ◆ **用户界面功能** 这些功能分别存储在特定于平台的 *Samples\UltraLite\CustDB* 子目录中。

通过完成以下步骤，可以将示例应用程序安装到运行 UltraLite 的远程设备上：

### ◆ 将示例应用程序安装到远程设备上

1. 启动统一数据库。
2. 启动 MobiLink 服务器。
3. 将示例应用程序安装到远程设备上。
4. 在远程设备上启动示例应用程序。
5. 同步示例应用程序。

### 示例

以下示例在运行 DB2 统一数据库的 Palm 设备上安装 CustDB 示例。

1. 确保统一数据库正在运行:

对于 DB2 数据库, 打开 DB2 命令窗口。键入以下命令, 其中 *userid* 和 *password* 是用于连接 DB2 数据库的用户 ID 和口令:

```
db2 connect to CustDB user userid using password
```

2. 启动 MobiLink 服务器:

对于 DB2 数据库, 在命令提示符下运行以下命令:

```
mksrv10 -c "DSN=CustDB" -zp
```

3. 将示例应用程序安装到 Palm 设备上:

- ◆ 在您的 PC 上启动 Palm Desktop。
- ◆ 在 [Palm Desktop] 工具栏上单击 [快速安装]。
- ◆ 单击 [添加]。浏览到 SQL Anywhere 10 安装目录的 *UltraLite\palm\68k* 子目录下的 *custdb.prc* 文件。
- ◆ 单击 [打开]。
- ◆ 对 Palm 设备执行 HotSync。

4. 在 Palm 设备上启动 CustDB 示例应用程序:

- ◆ 将 Palm 设备放入其底座中。

在第一次启动示例应用程序时, 会提示您进行同步, 以便下载数据的初始副本。仅当第一次启动该应用程序时才需要此步骤。在此之后, 下载的数据就存储在 UltraLite 数据库中。

- ◆ 启动示例应用程序。

在 [应用程序] 视图中, 单击 [CustDB]。

即会显示一个初始对话框, 提示您输入一个雇员 ID。

- ◆ 输入一个雇员 ID。

如果作为教程学习, 请输入值 50。示例应用程序还允许输入值 51、52 或 53, 但在这些情况下, 其行为略有不同。

有关每个用户 ID 的行为的详细信息, 请参见“[CustDB 示例中的用户](#)”一节第 57 页。

即会出现一个消息框, 通知您在继续操作前必须进行同步。

- ◆ 同步应用程序。

使用 HotSync 获取数据的初始副本。

- ◆ 确认数据已同步到应用程序中。

在 [应用程序] 视图中, 单击 [CustDB] 应用程序。屏幕将显示一个含有条目的客户登记表。

5. 在远程应用程序和统一数据库之间执行同步。当您已经更改了数据库时，仅需要完成该步骤。
  - ◆ 确保统一数据库和 MobiLink 服务器都在运行。
  - ◆ 将 Palm 设备放入其底座中。
  - ◆ 按 [HotSync] 按钮进行同步。

## CustDB 数据库中的表

CustDB 数据库的表定义位于 *samples-dir\MobiLink\CustDB* 目录的特定于平台的文件中。（有关 *samples-dir* 的信息，请参见“[示例目录](#)”一节《[SQL Anywhere 服务器 - 数据库管理](#)》。）

有关 CustDB 表的实体关系图，请参见“[关于 CustDB 示例数据库](#)”一节《[SQL Anywhere 10 - 简介](#)》。

统一数据库和远程数据库都包含以下五个表，不过这些表在两个数据库中的定义稍有不同。

### ULCustomer

ULCustomer 表包含一个客户列表。

在远程数据库中，ULCustomer 表包含下面几列：

- ◆ **cust\_id** 保存用于标识客户的唯一整数的主键列。
- ◆ **cust\_name** 长度为 30 个字符的字符串，其中包含客户名称。

在统一数据库中，ULCustomer 还包含以下附加列：

- ◆ **last\_modified** 包含上次修改该行的时间的时间戳。在进行基于时间戳的同步时将用到此列。

### ULProduct

ULProduct 表包含一个产品列表。

在远程数据库和统一数据库中，ULProduct 表包含以下几列：

- ◆ **prod\_id** 包含用于标识产品的唯一整数的主键列。
- ◆ **price** 标识单价的整数。
- ◆ **prod\_name** 长度为 30 个字符的字符串，其中包含产品名称。

### ULOrder

ULOrder 表包含一个订单列表，包括有关下订单的客户、接收订单的雇员以及所订购产品的详细信息。

在远程数据库中，ULOrder 表包含下面几列：

- ◆ **order\_id** 保存用于标识订单的唯一整数的主键列。
- ◆ **cust\_id** 引用 ULCustomer 的外键列。
- ◆ **prod\_id** 引用 ULProduct 的外键列。
- ◆ **emp\_id** 引用 ULEmployee 的外键列。
- ◆ **disc** 包含订单折扣的整数。
- ◆ **quant** 包含产品订购数量的整数。

- ◆ **notes** 长度为 50 个字符的字符串，包含订单的注释。
- ◆ **status** 长度为 20 个字符的字符串，用以描述订单的状态。

在统一数据库中，ULOrder 还包含以下附加列：

- ◆ **last\_modified** 包含上次修改该行的时间的戳。在进行基于时间戳的同步时将用到此列。

### ULOrderIDPool

ULOrderIDPool 表是 ULOrder 的主键池。

在远程数据库中，ULOrderIDPool 包含下面几列：

- ◆ **pool\_order\_id** 保存用于标识订单 ID 的唯一整数的主键列。

在统一数据库中，ULOrderIDPool 还包含以下附加列：

- ◆ **pool\_emp\_id** 一个整数列，包含订单 ID 被指派到的远程数据库的所有者的雇员 ID。
- ◆ **last\_modified** 包含上次修改该行的时间的戳。

### ULCustomerIDPool

ULCustomerIDPool 表是 ULCustomer 的主键池。

在远程数据库中，ULCustomerIDPool 包含下面几列：

- ◆ **pool\_cust\_id** 保存用于标识客户 ID 的唯一整数的主键列。

在统一数据库中，ULCustomerIDPool 还包含以下附加列：

- ◆ **pool\_emp\_id** 一个整数列，包含用于在远程数据库生成的新雇员的雇员 ID。
- ◆ **last\_modified** 包含上次修改该行的时间的戳。

下面的表仅包含在统一数据库中：

### ULIdentifyEmployee\_nosync

ULIdentifyEmployee\_nosync 表仅存在于统一数据库中。它包含如下所示的一列：

- ◆ **emp\_id** 该主键列包含一个代表雇员 ID 的整数。

### ULEmployee

ULEmployee 表仅存在于统一数据库中。它包含一个销售雇员列表。

ULEmployee 表包含下面几列：

- ◆ **emp\_id** 保存用于标识雇员的唯一整数的主键列。
- ◆ **emp\_name** 长度为 30 个字符的字符串，其中包含雇员名称。

### ULEmpCust

ULEmpCust 表控制将下载哪些客户的订单。如果雇员需要新客户的订单，则插入雇员 ID 和客户 ID 将会强制下载该客户的订单。

- ◆ **emp\_id** ULEmployee.emp\_id 的外键。
- ◆ **cust\_id** ULCustomer.cust\_id 的外键。主键包括 emp\_id 和 cust\_id。
- ◆ **action** 一个字符，用于确定是否应该从远程数据库中删除雇员记录。如果雇员不再需要客户的订单，则设置为 D（删除）。如果仍然需要订单，则设置为 NULL。

这种情况下必须使用逻辑删除，这样统一数据库能够识别从 ULOrder 表中删除哪些行。一旦下载完删除后，也可以从统一数据库中删除该雇员的 action 字段为 D 的所有记录。

- ◆ **last\_modified** 包含上次修改该行的时间的时间戳。在进行基于时间戳的同步时将用到此列。

### ULOldOrder 和 ULNewOrder

这些表仅存在于统一数据库中。它们用于解决冲突，所含的列与 ULOrder 相同。在 SQL Anywhere 和 Microsoft SQL Server 中，它们是临时表。在 Adaptive Server Enterprise 中，它们是普通表和 @@spid。DB2 和 Oracle 没有临时表，因此 MobiLink 需要能够识别哪些行属于同步用户。由于这些表都是基表，因此如果有五个用户正在同步，则他们每个人都同时在这些表中拥有一行。

有关 @@spid 的详细信息，请参见“变量”一节 [《SQL Anywhere 服务器 - SQL 参考》](#)。

## CustDB 示例中的用户

在 CustDB 示例中有两种类型的用户，销售人员和移动经理。二者的不同之处如下：

- ◆ **销售人员** 用户 ID 51、52 和 53 标识与销售人员相关的远程数据库。销售人员可以执行以下任务：
  - ◆ 查看客户列表和产品列表。
  - ◆ 添加新客户。
  - ◆ 添加或删除订单。
  - ◆ 滚动浏览未完成订单的列表。
  - ◆ 将所做更改同步到统一数据库。
- ◆ **移动经理** 用户 ID 50 标识与移动经理相关的远程数据库。移动经理可以执行与销售人员相同的任务。此外，移动经理还可以执行以下任务：
  - ◆ 接受或拒绝订单。

## 同步 CustDB

以下部分说明 CustDB 示例的同步逻辑。

### 同步逻辑源代码

可以使用 Sybase Central 检查统一数据库中的同步脚本。

#### 脚本类型与事件

通过调用 `ml_add_connection_script` 或 `ml_add_table_script`, `custdb.sql` 文件会将每个同步脚本都添加到统一数据库中。

#### 示例

`custdb.sql` 中的以下代码行将为 `ULProduct` 表添加一个表级别的脚本, 该脚本将在发生 `download_cursor` 事件期间执行。该脚本包含一个 `SELECT` 语句。

```
call ml_add_table_script(  
  'CustDB 10.0',  
  'ULProduct', 'download_cursor',  
  'SELECT prod_id, price, prod_name FROM ULProduct' )  
go
```

### 同步 CustDB 示例中的订单

#### 业务规则

`ULOrder` 表的业务规则如下:

- ◆ 只下载未批准或状态为空的订单。
- ◆ 在统一数据库和远程数据库中都可以修改订单。
- ◆ 每个远程数据库仅包含指派给雇员的订单。

#### 下载

可以在统一数据库中插入、删除, 或更新订单。在远程数据库中用来同步这些操作的脚本如下:

- ◆ **download\_cursor** `download_cursor` 脚本中的第一个参数是上次下载的时间戳。它用于确保仅下载那些自上次同步以来在远程数据库或统一数据库中进行过修改的行。第二个参数是雇员 ID。它用于确定下载哪些行。

CustDB 的 `download_cursor` 脚本如下所示:

```
CALL ULOrderDownload( {ml s.last_table_download}, {ml s.username} )
```

CustDB 的 `ULOrderDownload` 过程如下所示:

```
CREATE PROCEDURE ULOrderDownload ( IN LastDownload timestamp, IN  
  EmployeeID integer )  
BEGIN`
```

```

SELECT o.order_id, o.cust_id, o.prod_id, o.emp_id, o.disc, o.quant,
o.notes, o.status
  FROM ULOrder o, ULEmpCust ec
  WHERE o.cust_id = ec.cust_id
  AND ec.emp_id = EmployeeID
  AND ( o.last_modified >= LastDownload
  OR ec.last_modified >= LastDownload)
  AND ( o.status IS NULL OR o.status != 'Approved' )
  AND ( ec.action IS NULL )
END

```

- ◆ **download\_delete\_cursor** CustDB 的 download\_delete\_cursor 脚本如下所示:

```

SELECT o.order_id, o.cust_id, o.prod_id, o.emp_id, o.disc, o.quant, o.notes,
o.status
  FROM ULOrder o, dba.ULEmpCust ec
  WHERE o.cust_id = ec.cust_id
  AND ( ( o.status = 'Approved' AND o.last_modified >= {ml
s.last_table_download} )
  OR ( ec.action = 'D' ) )
  AND ec.emp_id = {ml s.username}

```

## 上载

可以在远程数据库中插入、删除或更新订单。在远程数据库中用来同步这些操作的脚本如下:

- ◆ **upload\_insert** CustDB 的 upload\_insert 脚本如下所示:

```

INSERT INTO ULOrder ( order_id, cust_id, prod_id, emp_id, disc, quant, notes,
status )
  VALUES( {ml r.order_id, r.cust_id, r.prod_id, r.emp_id, r.disc, r.quant,
r.notes, r.status } )

```

- ◆ **upload\_update** CustDB 的 upload\_update 脚本如下所示:

```

UPDATE ULOrder
  SET cust_id = {ml r.cust_id},
  prod_id = {ml r.prod_id},
  emp_id = {ml r.emp_id},
  disc = {ml r.disc},
  quant = {ml r.quant},
  notes = {ml r.notes},
  status = {ml r.status}
  WHERE order_id = {ml r.order_id}

```

- ◆ **upload\_delete** CustDB 的 upload\_delete 脚本如下所示:

```

DELETE FROM ULOrder WHERE order_id = {ml r.order_id}

```

- ◆ **upload\_fetch** CustDB 的 upload\_fetch 脚本如下所示:

```

SELECT order_id, cust_id, prod_id, emp_id, disc, quant, notes, status
  FROM ULOrder WHERE order_id = {ml r.order_id}

```

- ◆ **upload\_old\_row\_insert** CustDB 的 upload\_old\_row\_insert 脚本如下所示:

```

INSERT INTO ULOldOrder ( order_id, cust_id, prod_id, emp_id, disc, quant,
notes, status )

```

```
VALUES( {ml r.order_id, r.cust_id, r.prod_id, r.emp_id, r.disc, r.quant,  
r.notes, r.status } )
```

- ◆ **upload\_new\_row\_insert** CustDB 的 `upload_new_row_insert` 脚本如下所示:

```
INSERT INTO ULNewOrder ( order_id, cust_id, prod_id, emp_id, disc, quant,  
notes, status )  
VALUES( {ml r.order_id, r.cust_id, r.prod_id, r.emp_id, r.disc, r.quant,  
r.notes, r.status } )
```

### 冲突解决

- ◆ **resolve\_conflict** CustDB 的 `resolve_conflict` 脚本如下所示:

```
CALL ULResolveOrderConflict
```

CustDB 的 `ULResolveOrderConflict` 过程如下所示:

```
CREATE PROCEDURE ULResolveOrderConflict()  
BEGIN  
-- approval overrides denial  
IF 'Approved' = (SELECT status FROM ULNewOrder) THEN  
UPDATE ULOrder o  
SET o.status = n.status, o.notes = n.notes  
FROM ULNewOrder n  
WHERE o.order_id = n.order_id;  
END IF;  
DELETE FROM ULOldOrder;  
DELETE FROM ULNewOrder;  
END
```

## 同步 CustDB 示例中的客户

### 业务规则

用于管理客户的业务规则如下:

- ◆ 在统一数据库和远程数据库中都可以修改客户信息。
- ◆ 远程数据库和统一数据库中都包含一个完整的客户列表。

### 下载

可以在统一数据库中插入或更新客户信息。在远程数据库中用来同步这些操作的脚本如下:

- ◆ **download\_cursor** 以下 `download_cursor` 脚本下载自用户上次下载信息以来其信息发生更改的所有客户。

```
SELECT cust_id, cust_name FROM ULCustomer WHERE last_modified >= {ml  
s.last_table_download}
```

### 上载

您可以在远程数据库中插入、更新或删除客户信息。与这些操作相应的脚本如下:

- ◆ **upload\_insert** CustDB 的 `upload_insert` 脚本如下所示:

```
INSERT INTO ULCustomer( cust_id, cust_name )
VALUES( {ml r.cust_id, r.cust_name} )
```

- ◆ **upload\_update** CustDB 的 upload\_update 脚本如下所示:

```
UPDATE ULCustomer SET cust_name = {ml r.cust_name}
WHERE cust_id = {ml r.cust_id}
```

未在此表中执行冲突检测。

- ◆ **upload\_delete** CustDB 的 upload\_delete 脚本如下所示:

```
DELETE FROM ULCustomer WHERE cust_id = {ml r.cust_id}
```

## 同步 CustDB 示例中的产品

### 业务规则

为 ULProduct 下载所有行—这称为快照同步。

请参见“快照同步”一节《MobiLink - 服务器管理》。

ULProduct 表的业务规则如下:

- ◆ 只能在统一数据库中修改产品。
- ◆ 每个远程数据库中都包含所有产品。

### 下载

可以在统一数据库中插入、删除或更新产品信息。对应于这些操作的脚本如下所示:

- ◆ **download\_cursor** 以下 download\_cursor 脚本用于在每次同步时下载 ULProduct 表的所有行和列:

```
SELECT prod_id, price, prod_name FROM ULProduct
```

## 维护客户和订单的主键池

CustDB 示例数据库使用主键池来维护 ULCustomer 和 ULOrder 表中的唯一主键。主键池是 ULCustomerIDPool 和 ULOrderIDPool 表。

### ULCustomerIDPool

以下脚本在 ULCustomerIDPool 表中定义：

#### 下载

- ◆ **download\_cursor** CustDB 的 download\_cursor 脚本如下所示：

```
SELECT pool_cust_id FROM ULCustomerIDPool
WHERE last_modified >= {ml s.last_table_download}
AND pool_emp_id = {ml s.username}
```

#### 上载

- ◆ **upload\_insert** CustDB 的 upload\_insert 脚本如下所示：

```
INSERT INTO ULCustomerIDPool ( pool_cust_id )
VALUES( {ml r.pool_cust_id} )
```

- ◆ **upload\_delete** CustDB 的 upload\_delete 脚本如下所示：

```
DELETE FROM ULCustomerIDPool
WHERE pool_cust_id = {ml r.pool_cust_id}
```

- ◆ **end\_upload** 以下 end\_upload 脚本用于确保每次上载后客户 ID 池中保留 20 个客户 ID：

```
CALL ULOrderIDPool_maintain( {ml s.username} )
```

CustDB 的 UL\_CustomerIDPool\_maintain 过程如下所示：

```
CREATE PROCEDURE ULCustomerIDPool_maintain ( IN syncuser_id INTEGER )
BEGIN
  DECLARE pool_count INTEGER;
  -- Determine how many ids to add to the pool
  SELECT COUNT(*) INTO pool_count
  FROM ULCustomerIDPool
  WHERE pool_emp_id = syncuser_id;
  -- Top up the pool with new ids
  WHILE pool_count < 20 LOOP
    INSERT INTO ULCustomerIDPool ( pool_emp_id )
    VALUES ( syncuser_id );
    SET pool_count = pool_count + 1;
  END LOOP;
END
```

### ULOrderIDPool

以下脚本在 ULOrderIDPool 表中定义：

**下载**

- ◆ **download\_cursor** CustDB 的 download\_cursor 脚本如下所示:

```
SELECT pool_order_id FROM ULOrderIDPool
WHERE last_modified >= {ml s.last_table_download}
AND pool_emp_id = {ml s.username}
```

**上载**

- ◆ **end\_upload** 以下 end\_upload 脚本用于确保每次上载后订单 ID 池中保留 20 个订单 ID。

```
CALL ULOrderIDPool_maintain( {ml s.username} )
```

CustDB 的 UL\_OrderIDPool\_maintain 过程如下所示:

```
ALTER PROCEDURE ULOrderIDPool_maintain ( IN syncuser_id INTEGER )
BEGIN
  DECLARE pool_count INTEGER;
  -- Determine how many ids to add to the pool
  SELECT COUNT(*) INTO pool_count
  FROM ULOrderIDPool
  WHERE pool_emp_id = syncuser_id;
  -- Top up the pool with new ids
  WHILE pool_count < 20 LOOP
    INSERT INTO ULOrderIDPool ( pool_emp_id )
    VALUES ( syncuser_id );
    SET pool_count = pool_count + 1;
  END LOOP;
END
```

- ◆ **upload\_insert** CustDB 的 upload\_insert 脚本如下所示:

```
INSERT INTO ULOrderIDPool ( pool_order_id )
VALUES( {ml r.pool_order_id}
```

- ◆ **upload\_delete** CustDB 的 upload\_delete 脚本如下所示:

```
DELETE FROM ULOrderIDPool
WHERE pool_order_id = {ml r.pool_order_id}
```

## 清除

要重新启动示例，请重置 CustDB 数据库中的数据。

### ◆ 重置 CustDB 数据库中的数据

1. 在您的设备上安装 ULDBUtil:
  - ◆ 对于 Palm 设备，在您的 PC 上启动 Palm Desktop。
  - ◆ 在 [Palm Desktop] 工具栏上单击 [安装]。
  - ◆ 单击 [添加]。浏览到 SQL Anywhere 10 安装目录的 *UltraLite\palm\68k* 子目录下的 *uldbutil.prc* 文件。
  - ◆ 单击 [完成]。
  - ◆ 对 Palm 设备执行 HotSync。
2. 使用 ULDBUtil 删除数据:
  - ◆ 对于 Palm 设备，单击 [ULDBUtil] 图标。
  - ◆ 选择 [CustDB] 并单击 [删除数据]。
  - ◆ 对 Palm 设备执行 HotSync。

## 进一步阅读

有关脚本类型的详细信息，请参见“脚本类型”一节《MobiLink - 服务器管理》。

有关参考资料（包括每个脚本及其参数的详细信息），请参见“同步事件”《MobiLink - 服务器管理》。

---

---

## 第 4 章

# 研究 MobiLink Contact 示例

## 目录

Contact 示例教程简介 .....	68
Contact 示例安装 .....	69
Contact 数据库中的表 .....	71
Contact 示例中的用户 .....	73
同步 Contact 示例 .....	74
在 Contact 示例中监控统计信息和错误 .....	80

## Contact 示例教程简介

Contact 示例是 MobiLink 开发人员的宝贵资源。它以示例的形式说明如何实现开发 MobiLink 应用程序所需的多种技术。

Contact 示例应用程序中包含一个 SQL Anywhere 统一数据库和两个 SQL Anywhere 远程数据库。它阐释了几种常用的同步技术。为了从本章学到更多知识，请在阅读本章的同时学习该示例应用程序。

尽管统一数据库属于 SQL Anywhere 数据库，但同步脚本由 SQL 语句组成，而且这些语句应仅需要做少量更改即可应用于其它数据库管理系统。

Contact 示例在 *samples-dir\MobiLink>Contact* 中。有关概述，请参见同一位置的自述文件。（有关 *samples-dir* 的信息，请参见“[示例目录](#)”一节《[SQL Anywhere 服务器 - 数据库管理](#)》。）

### 同步设计

Contact 示例应用程序中的同步设计使用以下功能：

- ◆ **列子集** 统一数据库 Customer、Product、SalesRep 和 Contact 表中各列的子集与远程数据库共享。
- ◆ **行子集** 统一数据库中 SalesRep 表的所有列（但只有一行）与每个远程数据库共享。

请参见“[在远程数据库之间对行进行分区](#)”一节《[MobiLink - 服务器管理](#)》。

- ◆ **基于时间戳的同步** 这是一种用于识别自上次设备同步以来对统一数据库所做更改的方法。Customer、Contact 和 Product 表采用基于时间戳的方法进行同步。

请参见“[基于时间戳的下载](#)”一节《[MobiLink - 服务器管理](#)》。

## Contact 示例安装

您可以使用名为 *build.bat* 的 Windows 批处理文件来构建 Contact 示例数据库。在 UNIX 系统上，该文件是 *build.sh*。您最好检查该批处理文件的内容。它执行以下操作：

- ◆ 为统一数据库和每个远程数据库创建 ODBC 数据源定义。
- ◆ 创建一个名为 *consol.db* 的统一数据库，并将 MobiLink 系统表、数据库模式、一些数据、同步脚本及 MobiLink 用户名装载到数据库中。
- ◆ 在名为 *remote\_1* 和 *remote\_2* 的子目录下各创建一个远程数据库，都命名为 *remote.db*。装载两个数据库的共同信息，并应用自定义设置。这些自定义设置包括一个全局数据库标识符、一个 MobiLink 用户名及两个发布的预订。

### ◆ 构建 Contact 示例

1. 在命令提示符处，浏览至 *samples-dir\MobiLink\Contact*。
2. 运行 *build.bat* (Windows) 或 *build.sh* (Unix)。

有关 *samples-dir* 的信息，请参见“[示例目录](#)”一节《[SQL Anywhere 服务器 - 数据库管理](#)》。

## 运行 Contact 示例

Contact 示例中包含用于执行初始同步和演示 MobiLink 服务器及 *dbmlsync* 命令行的批处理文件。可以在文本编辑器中检查 *samples-dir\MobiLink\Contact* 中以下批处理文件的内容：

- ◆ *step1.bat*
- ◆ *step2.bat*
- ◆ *step3.bat*

### ◆ 运行 Contact 示例

1. 启动 MobiLink 服务器。

- ◆ 在命令提示符处，浏览至 *samples-dir\MobiLink\Contact* 并执行以下命令：

```
step1
```

该命令运行一个以详细模式启动 MobiLink 服务器的批处理文件。这种模式在开发或故障排除过程中非常有用，但由于其对性能影响很大，因此在日常生产环境中一般不采用。

2. 同步两个远程数据库。

- ◆ 在命令提示符处，浏览至 *samples-dir\MobiLink\Contact*。
- ◆ 执行下列命令：

step2

这是用于同步两个远程数据库的批处理文件。

3. 关闭 MobiLink 服务器。

- ◆ 在命令提示符处，浏览至 *samples-dir\MobiLink>Contact*。
- ◆ 执行以下命令：

step3

这是用于关闭 MobiLink 服务器的批处理文件。

要了解 Contact 示例中的同步方式，可以使用 Interactive SQL 修改远程数据库和统一数据库中的数据，并使用批处理文件进行同步。

## Contact 数据库中的表

Contact 数据库的表定义位于以下文件中（所有文件均位于示例目录下）：

- ◆ *MobiLink\Contact\build\_consol.sql*
- ◆ *MobiLink\Contact\build\_remote.sql*

统一数据库和远程数据库都包含以下三个表，不过这些表在两个数据库中的定义稍有不同。

### SalesRep

每个销售代表由 SalesRep 表中的一行来表示。每个远程数据库属于一个销售代表。

在每个远程数据库中，SalesRep 包含下面几列：

- ◆ **rep\_id** 包含销售代表标识号的主键列。
- ◆ **name** 销售代表的姓名。

在统一数据库中（远程数据库中没有），还有一个用于保存销售代表的 MobiLink 用户名的 ml\_username 列。

### Customer

此表为每个客户保存一行信息。每个客户即是一个与某一销售代表有业务关系的公司。在表 SalesRep 和 Customer 之间存在一对多的关系。

在每个远程数据库中，Customer 包含下面几列：

- ◆ **cust\_id** 用于保存客户标识号的主键列。
- ◆ **name** 客户名。此名称为公司名称。
- ◆ **rep\_id** 引用 SalesRep 表的外键列。标识指派给该客户的销售代表。

在统一数据库中，有两个附加列：last\_modified 和 active：

- ◆ **last\_modified** 行的最近一次修改时间。在进行基于时间戳的同步时将用到此列。
- ◆ **active** 用于说明客户当前状态（处于活动状态 (1) 或不再与公司有业务往来 (0)）的 BIT 列。如果此列标记为非活动状态 (0)，则所有与此客户相关的行将从远程数据库中删除。

### Contact

此表为每个联系人保存一行信息。联系人是客户公司的员工。表 Customer 和 Contact 之间存在一对多的关系。

在每个远程数据库中，Contact 包含下面几列：

- ◆ **contact\_id** 用于保存联系人标识号的主键列。
- ◆ **name** 各联系人的姓名。

- ◆ **cust\_id** 联系人所属客户的标识符。

在统一数据库中，该表还包含以下列：

- ◆ **last\_modified** 行的最近一次修改时间。在进行基于时间戳的同步时将用到此列。
- ◆ **active** 用于说明联系人当前状态（处于活动状态 (1) 或不再与公司有业务往来 (0)）的 BIT 列。如果此列标记为非活动状态 (0)，则与此联系人相关的行将从远程数据库中删除。

## Product

在 Product 表中，公司所销售的每种产品占一行。Product 表在单独的发布中保存，因此远程数据库可以单独对该表进行同步。

在每个远程数据库中，Product 包含下面几列：

- ◆ **id** 包含产品标识号的主键列。
- ◆ **name** 产品的名称。
- ◆ **size** 产品的大小。
- ◆ **quantity** 产品的库存数量。在销售代表获得订单时，此列会被更新。
- ◆ **unit\_price** 产品的单价。

在统一数据库中，Product 表还包含以下附加列：

- ◆ **supplier** 产品的制造商。
- ◆ **last\_modified** 行的最近一次修改时间。在进行基于时间戳的同步时将用到此列。
- ◆ **active** 用于说明产品当前是否处于活动状态 (1) 的 BIT 列。如果此列标记为非活动状态 (0)，则与此产品相关的行将从远程数据库中删除。

除上述这些表以外，在统一数据库中还将创建一组表。其中包括 **product\_conflict** 表（在解决冲突时使用的临时表）和一组属于用户 **mlmaint** 的用于监控 MobiLink 活动的表。用来创建 MobiLink 监控表的脚本位于 *samples-dir\MobiLink>Contact\mlmaint.sql* 文件中。

## Contact 示例中的用户

在 Contact 示例中包含多个不同的数据库用户 ID 和 MobiLink 用户名。

### 数据库用户 ID

两个远程数据库分别属于销售代表 Samuel Singer (rep\_id 856) 和 Pamela Savarino (rep\_id 949)。

在连接其远程数据库时，这两个用户都使用缺省 SQL Anywhere 用户 ID `dba` 及口令 `SQL`。

每个远程数据库中还都有一个用户 ID `sync_user`，其口令为 `sync_user`。此用户 ID 只在 `dbmlsync` 命令行中使用。该用户拥有 `REMOTE DBA` 权限，因此如果从 `dbmlsync` 连接，该用户可以执行任何操作，但如果从其它应用程序连接，该用户则没有任何权限。因此，广泛使用该用户 ID 及口令并不会出现问题。

在统一数据库中，有一个名为 `mlmaint` 的用户，该用户拥有用于监控 MobiLink 同步统计数据及错误的表。但此用户没有进行连接的权限。这种将表指派给单独用户 ID 的做法仅仅是为了在模式中将某些对象与其它对象分开，以便在 Sybase Central 及其它实用程序中进行管理。

### MobiLink 用户名

MobiLink 用户名与数据库用户 ID 不同。除了在连接数据库时所使用的用户 ID 以外，每个远程设备还拥有一个 MobiLink 用户名。Samuel Singer 的 MobiLink 用户名为 `SSinger`。Pamela Savarino 的 MobiLink 用户名为 `PSavarino`。MobiLink 用户名在以下位置储存或使用：

- ◆ 在远程数据库中，使用 `CREATE SYNCHRONIZATION USER` 语句添加 MobiLink 用户名。
- ◆ 在统一数据库中，使用 `mluser` 实用程序添加 MobiLink 用户名及口令。
- ◆ 在同步过程中，正在连接的用户的 MobiLink 口令在 `MobiLink\Contact\step2.bat` 所列 `dbmlsync` 命令行中提供。
- ◆ MobiLink 服务器在同步过程中将 MobiLink 用户名作为参数提供给许多脚本。
- ◆ 统一数据库的 `SalesRep` 表中包含一个 `ml_username` 列。同步脚本会将 MobiLink 用户名参数与此列中的值相比较。

## 同步 Contact 示例

下面的部分说明 Contact 示例的同步逻辑。

### 同步 Contact 示例中的销售代表

SalesRep 表的同步脚本演示了何为**快照同步**。销售代表的信息无论是否发生更改都要进行下载。请参见“快照同步”一节《MobiLink - 服务器管理》。

#### 业务规则

SalesRep 表的业务规则如下：

- ◆ 不能在远程数据库中修改该表。
- ◆ 不能更改销售代表的 MobiLink 用户名及 rep\_id 值。
- ◆ 每个远程数据库只包含 SalesRep 表中与远程数据库所有者的 MobiLink 用户名相对应的一行。

#### 下载

- ◆ **download\_cursor** 在每个远程数据库中，SalesRep 表包含单独一行。下载单独一行需要的开销非常小，因此使用一个简单的快照 download\_cursor 脚本：

```
SELECT rep_id, name
FROM SalesRep
WHERE ? IS NOT NULL
AND ml_username = ?
```

脚本中的第一个参数为最后一次下载时间戳（此处未使用）。IS NOT NULL 表达式是一个虚表达式，用以使用该参数。第二个参数为 MobiLink 用户名。

#### 上载

此表不应在远程数据库中更新，所以没有针对此表的上载脚本。

### 同步 Contact 示例中的客户

Customer 表的同步脚本演示了**基于时间戳的同步**及行的分区。这两种技术既维护了表数据的一致性，又最大程度减小了同步期间所传送的数据量。

请参见：

- ◆ “基于时间戳的下载”一节《MobiLink - 服务器管理》
- ◆ “在远程数据库之间对行进行分区”一节《MobiLink - 服务器管理》

#### 业务规则

用于管理客户的业务规则如下：

- ◆ 在远程数据库和统一数据库中都可以修改客户信息。

- ◆ 客户将定期在销售代表之间重新指派。此过程通常称为地域调整。
- ◆ 每个远程数据库中仅包含所指派的客户。

## 下载

- ◆ **download\_cursor** 以下 `download_cursor` 脚本只下载自上次成功下载以来信息已经发生更改的活动客户。它还可以按销售代表过滤客户。

```
SELECT cust_id, Customer.name, Customer.rep_id
FROM Customer key join SalesRep
WHERE Customer.last_modified >= ?
AND SalesRep.ml_username = ?
AND Customer.active = 1
```

- ◆ **download\_delete\_cursor** 以下 `download_delete_cursor` 脚本只下载自上次成功下载以来信息已经发生更改的客户。它会删除标记为非活动和未指派给销售代表的客户。

```
SELECT cust_id
FROM Customer key join SalesRep
WHERE Customer.last_modified >= ?
AND ( SalesRep.ml_username != ? OR Customer.active = 0 )
```

如果从统一数据库 `Customer` 表中删除了某些行，它们将不会出现在此结果集中，因此也不会从远程数据库中删除。但这些客户会被标记为非活动状态。

进行地域调整后，此脚本将删除不再指派给该销售代表的客户。它还会删除转让给其他销售代表的客户。这些附加删除将被标志为 `SQLCODE 100`，但并不妨碍同步过程。可以编写更复杂的脚本来只识别那些从当前销售代表转让出去的客户。

MobiLink 客户端将在远程数据库上执行级联删除，因此该脚本还将删除所有为指派给其他销售代表的客户工作的联系人。

## 上载

您可以在远程数据库中插入、更新或删除客户信息。与这些操作相应的脚本如下：

- ◆ **upload\_insert** 以下 `upload_insert` 脚本将在 `Customer` 表中添加一行，并将客户标记为活动：

```
INSERT INTO Customer(
    cust_id, name, rep_id, active )
VALUES ( ?, ?, ?, 1 )
```

- ◆ **upload\_update** 以下 `upload_update` 脚本将在统一数据库中修改客户信息。不在此表中执行冲突检测。

```
UPDATE Customer
SET name = ?, rep_id = ?
WHERE cust_id = ?
```

- ◆ **upload\_delete** 以下 `upload_delete` 脚本在统一数据库中将客户标记为非活动。但它不删除行。

```
UPDATE Customer
SET active = 0
WHERE cust_id = ?
```

## 同步 Contact 示例中的联系人

Contact 表中包含在客户公司工作的员工的姓名、指向该客户的外键和一个用于标识该联系人的唯一整型值。它还包含一个 last\_modified 时间戳和用于指示该联系人是否为活动状态的标记。

### 业务规则

此表的业务规则如下：

- ◆ 在统一数据库和远程数据库中都可以修改联系人信息。
- ◆ 每个远程数据库只包含为所指派客户工作的联系人。
- ◆ 在销售代表间重新指派客户时，联系人也必须重新指派。

### 触发器

Customer 表中的触发器用于确保在客户信息发生更改时联系人也执行相应的更改。在客户发生更改时，触发器将显式地更改其每个联系人的 last\_modified 列：

```
CREATE TRIGGER UpdateCustomerForContact
AFTER UPDATE OF rep_id ORDER 1
ON DBA.Customer
REFERENCING OLD AS old_cust NEW as new_cust
FOR EACH ROW
BEGIN
    UPDATE Contact
    SET Contact.last_modified = new_cust.last_modified
    FROM Contact
    WHERE Contact.cust_id = new_cust.cust_id
END
```

触发器在客户发生修改时会更新所有联系人记录，使客户与其相关联的联系人结合在一起。只要客户发生修改，所有关联联系人也会随之修改，且客户和关联联系人会在下一次同步过程中一同下载。

### 下载

- ◆ **download\_cursor** Contact 的 download\_cursor 脚本如下所示：

```
SELECT contact_id, contact.name, contact.cust_id
FROM ( contact_JOIN customer ) JOIN salesrep
ON contact.cust_id = customer.cust_id
   AND customer.rep_id = salesrep.rep_id
WHERE Contact.last_modified >= ?
   AND salesrep.ml_username = ?
   AND Contact.active = 1
```

此脚本将检索指派给此代表的、自销售代表上次下载（显式下载或因相应客户的修改而引发的下载）以后更改过的活动联系人。表 Customer 和 SalesRep 之间需要有一个连接来识别与该销售代表相关联的联系人。

- ◆ **download\_delete\_cursor** Contact 的 download\_delete\_cursor 脚本如下所示：

```
SELECT contact_id
FROM ( Contact_JOIN Customer ) JOIN SalesRep
ON Contact.cust_id = Customer.cust_id
   AND Customer.rep_id = SalesRep.rep_id
```

```
WHERE Contact.last_modified >= ?
AND Contact.active = 0
```

由于 MobiLink 客户端将自动使用级联参照完整性，因此在从远程数据库中删除客户时，相应的联系人也将删除。所以 `download_delete_cursor` 脚本必须仅删除标记为非活动的联系人。

## 上载

可以在远程数据库中插入、更新或删除联系人信息。与这些操作相应的脚本如下：

- ◆ **upload\_insert** 以下 `upload_insert` 脚本将在 `Contact` 表中添加一行，并将联系人标记为活动：

```
INSERT INTO Contact (
    contact_id, name, cust_id, active )
VALUES ( ?, ?, ?, 1 )
```

- ◆ **upload\_update** 以下 `upload_update` 脚本将在统一数据库中修改联系人信息：

```
UPDATE Contact
SET name = ?, cust_id = ?
WHERE contact_id = ?
```

未在此表中执行冲突检测。

- ◆ **upload\_delete** 以下 `upload_delete` 脚本在统一数据库中将联系人标记为非活动。但它不删除行。

```
UPDATE Contact
SET active = 0
WHERE contact_id = ?
```

## 同步 Contact 示例中的产品

`Product` 表的脚本演示了冲突检测及其解决办法。

`Product` 表与其它表分别保存在不同的发布中，因此它可以单独下载。例如：如果价格发生变化而销售代表正在通过速度缓慢的链接进行同步，他们无需上载自己客户和联系人的更改即可下载相关产品的更改。

## 业务规则

远程数据库中可以进行唯一更改是在获取订单时更改数量列。

## 下载

- ◆ **download\_cursor** 以下 `download_cursor` 脚本将下载自从上次远程数据库同步后发生更改的所有行：

```
SELECT id, name, size, quantity, unit_price
FROM product
WHERE last_modified >= ?
AND active = 1
```

- ◆ **download\_delete\_cursor** 以下 `download_delete_cursor` 脚本删除公司停止销售的所有产品。这些产品在统一数据库中被标记为非活动状态。

```
SELECT id, name, size, quantity, unit_price
FROM product
WHERE last_modified >= ?
AND active = 0
```

## 上载

从远程数据库只上载 UPDATE 操作。这些上载脚本的主要功能是冲突检测与解决过程。

如果两个销售代表获取订单并执行同步，则每个订单都将从 Product 表的数量列中减去。例如，如果 Samuel Singer 获得了一个 20 套棒球帽（产品 ID 400）的订单，他会将数量列从 90 更改为 70；如果 Pamela Savarino 在接收到此更改之前获得了一个 10 套棒球帽的订单，她会将自己数据库中的数量列从 90 更改为 80。

因此在 Samuel Singer 同步他的更改时，统一数据库中的数量列由 90 变为 70，而当 Pamela Savarino 同步她的更改时，正确的操作应该是将值设置为 60。而这一设置就是由冲突检测完成的。

冲突检测模式中包含以下脚本：

- ◆ **upload\_update** 以下 upload\_update 脚本是统一数据库中的一个简单的 UPDATE：

```
UPDATE product
SET name = ?, size = ?, quantity = ?, unit_price = ?
WHERE product.id = ?
```

- ◆ **upload\_fetch** 以下 upload\_fetch 脚本从 Product 表中读取一个单独的行，并与已上载的行的旧值进行比较。如果这两行的值不同，则说明检测到冲突。

```
SELECT id, name, size, quantity, unit_price
FROM Product
WHERE id = ?
```

- ◆ **upload\_old\_row\_insert** 如果检测到冲突，则将旧值放入 product\_conflict 表中，以供 resolve\_conflict 脚本使用。添加该行时将在 row\_type 列写入 O（表示原值）。

```
INSERT INTO DBA.product_conflict(
    id, name, size, quantity, unit_price, row_type )
VALUES( ?, ?, ?, ?, ?, 'O' )
```

- ◆ **upload\_new\_row\_insert** 以下脚本将在 product\_conflict 表中添加已上载的行的新值，以供 resolve\_conflict 脚本使用：

```
INSERT INTO DBA.product_conflict(
    id, name, size, quantity, unit_price, row_type )
VALUES( ?, ?, ?, ?, ?, 'N' )
```

## 冲突解决

- ◆ **resolve\_conflict** 以下脚本通过将旧行和新行之间的差加到统一数据库的数量值中来解决冲突：

```
UPDATE Product
SET p.quantity = p.quantity
    - old_row.quantity
    + new_row.quantity
FROM Product p,
    DBA.product_conflict old_row,
    DBA.product_conflict new_row
WHERE p.id = old_row.id
```

```
AND p.id = new_row.id  
AND old_row.row_type = 'O'  
AND new_row.row_type = 'N'
```

## 在 Contact 示例中监控统计信息和错误

Contact 示例中包含一些简单的错误报告及监控脚本。用来创建这些脚本的 SQL 语句位于 *MobiLink \Contact\mlmaint.sql* 文件中。

这些脚本将行插入用于保存这些值的表中。为了方便起见，这些表属于特别的用户 *mlmaint*。

---

## 第 5 章

# 教程：将 MobiLink 用于 Oracle 10g 统一数据库

## 目录

MobiLink Oracle 教程简介 .....	82
第 1 课：创建数据库 .....	83
第 2 课：启动 MobiLink 服务器 .....	88
第 3 课：启动 MobiLink 同步客户端 .....	89
进一步阅读 .....	90

## MobiLink Oracle 教程简介

在本教程中，需要准备一个 Oracle 统一数据库和一个 SQL Anywhere 远程数据库。

### 必需的软件

- ◆ SQL Anywhere 的完全安装。
- ◆ Oracle Enterprise Edition 10g 的完全安装。
- ◆ iAnywhere Solutions - Oracle 驱动程序

### 能力和经验

在学习本教程前，您应具备以下能力及经验：

- ◆ 熟悉 Sybase Central 接口和功能。
- ◆ 熟悉 Interactive SQL 和 Oracle SQL Plus。
- ◆ 能够胜任 Oracle 编程。

### 目标

本教程的目标是：

- ◆ 熟悉用于 Oracle 的 MobiLink 服务器及相关组件。
- ◆ 掌握与 Oracle 统一数据库相关的 MobiLink 服务器和客户端命令的执行。

特别是，您将执行下面的任务：

- ◆ 创建新的 SQL Anywhere 数据库作为远程数据库。
- ◆ 启动使用 Oracle 统一数据库的 MobiLink 服务器。
- ◆ 启动 MobiLink 同步客户端，并实现了远程数据库和 Oracle 统一数据库的同步。

### 建议阅读的背景知识

有关运行 MobiLink 服务器的详细信息，请参见“[MobiLink 同步简介](#)”第 3 页。

## 第 1 课：创建数据库

MobiLink 同步要求您有一个统一数据库（本教程中为 Oracle）、一个远程数据库（本教程中为 SQL Anywhere），且这两个数据库中的每个数据库都要有一个 ODBC 数据源。

### 创建 SQL Anywhere 远程数据库

#### ◆ 创建远程数据库

1. 为本教程创建一个目录（例如，*OracleTut*）。打开命令提示符，导航到 *OracleTut*。
2. 键入以下命令：

```
dbinit remote.db
```

3. 通过列出 *OracleTut* 的内容来验证数据库创建是否成功。

#### ◆ 为远程数据库创建 ODBC 数据源

- 在仍位于 *OracleTut* 目录时，键入以下命令（全部在一行上）：

```
dbdsn -w test_remote -y  
-c "uid=DBA;pwd=sql;dbf=path\OracleTut\remote.db;eng=remote"
```

请用您的 *OracleTut* 目录位置替换 *path*。

#### ◆ 在远程数据库中创建对象

1. 启动 Interactive SQL。  
选择 [开始] ► [程序] ► [SQL Anywhere 10] ► [Interactive SQL]。
2. 连接到远程数据库。
3. 创建远程表、发布、用户和预订：

将以下代码复制到 Interactive SQL 中并执行它。

```
CREATE TABLE emp ( emp_id int primary key ,emp_name varchar( 128 ) );  
CREATE TABLE cust(  
    cust_id int primary key,  
    emp_id int references emp ( emp_id ),  
    cust_name varchar( 128 ) );  
CREATE PUBLICATION emp_cust ( TABLE cust, TABLE emp );  
CREATE SYNCHRONIZATION USER ml_user;  
CREATE SYNCHRONIZATION SUBSCRIPTION  
    TO emp_cust FOR ml_user TYPE TCPIP ADDRESS 'host=localhost';
```

#### 进一步阅读

有关创建 SQL Anywhere 数据库的详细信息，请参见“初始化实用程序 (dbinit)”一节《SQL Anywhere 服务器 - 数据库管理》。

有关创建 SQL Anywhere 数据库的 ODBC 数据源的详细信息，请参见“数据源实用程序 (dbdsn)”一节《SQL Anywhere 服务器 - 数据库管理》。

有关 Interactive SQL 的详细信息，请参见“[Interactive SQL 实用程序 \(dbisql\)](#)”一节《[SQL Anywhere 服务器 - 数据库管理](#)》。

有关创建本教程中 SQL Anywhere 对象的信息，请参见：

- ◆ “[CREATE TABLE 语句](#)”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 SYNCHRONIZATION USER 语句 \[MobiLink\]](#)”一节《[SQL Anywhere 服务器 - SQL 参考](#)》
- ◆ “[CREATE SYNCHRONIZATION SUBSCRIPTION 语句 \[MobiLink\]](#)”一节《[SQL Anywhere 服务器 - SQL 参考](#)》

## 创建 Oracle 统一数据库

您可以使用多种方法将数据输入到 Oracle 数据库中。本教程使用 Oracle SQL Plus。

### ◆ 创建 Oracle 统一数据库

1. 启动 SQL Plus。

选择 [开始] ► [程序] ► [Oracle - OraDb10g\_home1] ► [Application Development] ► [SQL Plus]。

2. 连接到统一数据库。

3. 将以下代码复制到 SQL Plus 中并执行它。这些 SQL 语句在统一数据库中删除、创建和填充表。如果要删除的表不存在，SQL Plus 输出中将出现一个错误，但此错误不影响处理。

```
CREATE SEQUENCE emp_sequence;
CREATE SEQUENCE cust_sequence;
DROP TABLE emp;
CREATE TABLE emp(
    emp_id int primary key,
    emp_name varchar( 128 ) );
DROP TABLE cust;
CREATE TABLE cust(
    cust_id int primary key,
    emp_id int references emp(emp_id),
    cust_name varchar( 128 ) );
INSERT INTO emp
    ( emp_id, emp_name ) VALUES ( emp_sequence.nextval, 'emp1' );
INSERT INTO emp
    ( emp_id, emp_name ) VALUES ( emp_sequence.nextval, 'emp2' );
INSERT INTO emp
    ( emp_id, emp_name ) VALUES ( emp_sequence.nextval, 'emp3' );
COMMIT;
INSERT INTO cust
    ( cust_id, emp_id, cust_name ) VALUES ( cust_sequence.nextval, 1,
'cust1' );
INSERT INTO cust
    ( cust_id, emp_id, cust_name ) VALUES ( cust_sequence.nextval, 1,
'cust2' );
INSERT INTO cust
    ( cust_id, emp_id, cust_name ) VALUES ( cust_sequence.nextval, 2,
```

```
'cust3' );  
COMMIT;
```

4. 要验证表是否已成功创建，为每个表执行以下 SQL 语句：

```
SELECT * FROM emp;  
SELECT * FROM cust;
```

保持统一数据库处于运行状态。

### 为统一数据库创建 ODBC 数据源

MobiLink 需要通过 ODBC 数据源执行数据同步。对于版本 10.0.0，必须下载 Oracle ODBC 驱动程序。有关信息，请参见 [http://www.ianywhere.com/developer/technotes/odbc\\_mobilink.html](http://www.ianywhere.com/developer/technotes/odbc_mobilink.html)。

您必须知道实例、服务和数据库的名称，因为在安装过程的 ODBC 部分需要用到这些值。这些值在安装 Oracle 时确立。

以下步骤为 Oracle 统一数据库设置 ODBC 配置。

#### ◆ 为 Oracle 建立 ODBC 数据源：

1. 选择 [开始] ► [程序] ► [SQL Anywhere 10] ► [SQL Anywhere] ► [ODBC 管理器]。  
将打开 [ODBC 数据源管理器]。
2. 单击 [用户 DSN] 选项卡上的 [添加]。[创建新数据源] 窗口随即出现。
3. 选择 [iAnywhere Solutions 10 - Oracle] 并单击 [完成]。  
将出现 [ODBC Oracle 驱动程序安装] 窗口。
4. 单击 [常规] 选项卡，然后键入数据源名称 **ora\_consol**。此名称是用于连接到 Oracle 数据库的 DSN。稍后您将用到它。
5. 输入服务器名称。此值取决于 Oracle 服务器安装在哪台计算机上。如果服务器安装在您的计算机上，可将此字段保留为空。
6. 单击 [高级] 选项卡，然后输入一个默认用户名。在本教程中，可以使用 **system**，或任何具有足够的权限可以创建对象的用户名。
7. 单击 [确定]。
8. 单击 [确定]，关闭 [ODBC 数据源管理器]。

### 进一步阅读

有关 Oracle ODBC 驱动程序的详细信息，请参见“[iAnywhere Solutions Oracle 驱动程序](#)”一节《[MobiLink - 服务器管理](#)》。

有关 Oracle 的详细信息，请参见 Oracle 文档。

## 建立统一数据库

与 MobiLink 一起提供了一个名为 *syncora.sql* 的脚本，该脚本位于 SQL Anywhere 安装目录的 *MobiLink\setup* 子目录下。运行此脚本来建立使用 MobiLink 的 Oracle 数据库。

*Syncora.sql* 包含用 Oracle SQL 编写的 SQL 语句，用于准备 Oracle 数据库，以便将其用作统一数据库。该文件还将创建一系列 MobiLink 系统表、触发器和过程供 MobiLink 使用。这些系统表以 *ML\_* 为前缀。MobiLink 将在同步过程中使用这些表。

### ◆ 在 Oracle 中创建 MobiLink 系统表

1. 启动 SQL Plus。选择 [开始] ► [程序] ► [Oracle - OraDb10g\_home1] ► [Application Development] ► [SQL Plus]。

使用 Oracle SQL Plus 连接到您的 Oracle 数据库。使用 **system** 模式登录，口令为 **manager**。

2. 通过键入以下命令，运行 *syncora.sql*：

```
@path\syncora.sql;
```

其中，*path* 为 SQL Anywhere 10 安装目录的 *MobiLink\setup* 子目录。如果路径中有空格，则使用引号将路径和文件名引起来。

### ◆ 验证系统表已安装

1. 启动 SQL Plus。选择 [开始] ► [程序] ► [Oracle - OraDb10g\_home1] ► [Application Development] ► [SQL Plus]。

2. 运行以下 SQL 语句，列出 MobiLink 系统表、过程和触发器：

```
SELECT object_name  
FROM all_objects  
WHERE object_name  
LIKE 'ML_%';
```

如果没有以 *ML\_* 开头的对象，则表示您刚刚执行的操作不成功。在这种情况下，需要检查 MobiLink 错误消息，寻找问题所在；解决问题；然后按照以下方法删除 MobiLink 系统表。

### ◆ 删除 MobiLink 系统表（如果必要）

1. 在 SQL Plus 中运行以下 SQL 语句：

```
select 'drop ' || object_type || ' ' || object_name || ';'   
from all_objects   
where object_name like 'ML_%';
```

此语句将生成要删除的表、过程和触发器的列表。

2. 将该列表复制到一个文本文件中，然后在您的 *OracleTut* 目录中将其保存为 *drop.sql*。删除所有不包含 **DROP** 语句的行。

3. 运行以下命令，执行 *drop.sql* 中的 SQL 语句：

```
@path\OracleTut\drop.sql;
```

请用您的 *OracleTut* 目录位置替换 *path*。再次运行 *drop.sql*，删除第一次运行时由于依存性而未删除的表。

您现在可以重新按照上述在 Oracle 中创建 MobiLink 系统表的说明进行操作。

有关建立 Oracle 统一数据库的详细信息，请参见“Oracle 统一数据库”一节《MobiLink - 服务器管理》。

## 第 2 课：启动 MobiLink 服务器

现在可以从命令提示符启动 MobiLink 服务器了。由于 MobiLink 服务器是统一数据库的客户端，因此必须在启动 MobiLink 服务器之前先启动统一数据库。

### ◆ 启动 MobiLink 服务器

1. 确保统一数据库正在运行。
2. 在命令提示符下，转到 *OracleTut* 目录。
3. 启动 MobiLink 服务器。

键入以下命令：

```
mlsrv10 -c "dsn=ora_consol;pwd=manager;uid=system" -o mlsrv.mls -v+ -zu+
```

此命令行指定以下 mlsrv10 选项：

- ◆ **-c** 指定连接参数。注意，因为 DSN 包含用户 ID，所以上示例只指定口令。请参见“[-c 选项](#)”一节《[MobiLink - 服务器管理](#)》。
- ◆ **-o** 指定消息日志文件。请参见“[-o 选项](#)”一节《[MobiLink - 服务器管理](#)》。
- ◆ **-v+** 启用详细记录。请参见“[-v 选项](#)”一节《[MobiLink - 服务器管理](#)》。
- ◆ **-dl** 将显示日志功能设置为 ON。请参见“[-dl 选项](#)”一节《[MobiLink - 服务器管理](#)》。
- ◆ **-zu+** 自动化用户验证过程。请参见“[-zu 选项](#)”一节《[MobiLink - 服务器管理](#)》。

### 进一步阅读

有关 mlsrv10 的详细信息，请参见“[MobiLink 服务器](#)”《[MobiLink - 服务器管理](#)》和“[mlsrv10 语法](#)”一节《[MobiLink - 服务器管理](#)》。

## 第 3 课：启动 MobiLink 同步客户端

现在可以从命令提示符启动 MobiLink 客户端了。MobiLink 客户端启动同步。

您可以使用 `-c` 选项在 `dbmlsync` 命令行中指定远程数据库的连接参数。

### ◆ 启动 MobiLink 客户端

1. 确保 MobiLink 服务器已启动。
2. 在命令提示符下，转到 *OracleTut* 目录。
3. 键入以下命令：

```
dbmlsync -c "dsn=test_remote" -o dbmlsync.out -v+ -e "SendColumnNames=ON"
```

此命令行指定以下 `dbmlsync` 选项：

- ◆ `-c` 提供数据库连接参数。请参见“`-c` 选项”一节《MobiLink - 客户端管理》。
- ◆ `-o` 指定消息日志文件。请参见“`-o` 选项”一节《MobiLink - 客户端管理》。
- ◆ `-v+` 详细操作。请参见“`-v` 选项”一节《MobiLink - 客户端管理》。
- ◆ `-e` 扩展选项。指定 "SendColumnNames=ON" 将把列名发送到 MobiLink。请参见“MobiLinkSQL Anywhere 客户端扩展选项”《MobiLink - 客户端管理》。

### 进一步阅读

有关 `dbmlsync` 的详细信息，请参见“`dbmlsync` 语法”一节《MobiLink - 客户端管理》。

## 进一步阅读

有关运行 MobiLink 服务器的详细信息，请参见“[MobiLink 服务器](#)”《[MobiLink - 服务器管理](#)》。

有关编写同步脚本的详细信息，请参见“[编写同步脚本](#)”《[MobiLink - 服务器管理](#)》和“[同步事件](#)”《[MobiLink - 服务器管理](#)》。

有关其它同步方法（如时间戳）的介绍，请参见“[同步技术](#)”《[MobiLink - 服务器管理](#)》。

---

## 第 6 章

# 教程：使用 Java 同步逻辑

## 目录

Java 同步教程简介 .....	92
第 1 课：编译 CustdbScripts Java 类 .....	93
第 2 课：指定类方法来处理事件 .....	95
第 3 课：用 -sl java 运行 MobiLink 服务器 .....	98
第 4 课：测试同步 .....	99
清除 .....	100
进一步阅读 .....	101

## Java 同步教程简介

本教程将指导您完成使用 Java 同步逻辑的基本步骤。可使用 CustDB 示例作为 SQL Anywhere 统一数据库，为 MobiLink 表级别事件指定简单的类方法。该过程还涉及使用用来设置编译的 Java 类路径的选项来运行 MobiLink 服务器 (mlsrv10)。

### 必需的软件

- ◆ SQL Anywhere 10.0
- ◆ Java 软件开发工具包

### 能力和经验

您需要：

- ◆ 熟悉 Java
- ◆ 了解 MobiLink 事件脚本的基本知识

### 目标

您将掌握并熟悉以下内容：

- ◆ 使用简单的 Java 类方法编写 MobiLink 表级别事件

### 重要概念

本节使用以下步骤通过 MobiLink CustDB 示例数据库实现基本的基于 Java 的同步：

- ◆ 使用 MobiLink 服务器 API 引用编译源文件
- ◆ 为特定表级别事件指定类方法
- ◆ 用 -sl java 选项运行 MobiLink 服务器 (mlsrv10)
- ◆ 使用示例 Windows 客户端应用程序测试同步

### 建议阅读的背景知识

有关同步脚本的详细信息，请参见“同步脚本介绍”一节 [《MobiLink - 服务器管理》](#)。

## 第 1 课：编译 CustdbScripts Java 类

Java 类能够以多种方法封装同步逻辑。

在本课中，您将编译一个与 CustDB 示例数据库关联的类。

### MobiLink 数据库示例

SQL Anywhere 随附一个 SQL Anywhere 示例数据库 (CustDB)，该数据库已进行了同步设置，其中包括同步所需的 SQL 脚本。例如，CustDB ULCustomer 表是一个同步表，它支持多种表级别事件。

CustDB 的设计用途是作为 UltraLite 和 SQL Anywhere 客户端的统一数据库服务器。CustDB 数据库有一个名为 SQL Anywhere 10 CustDB 的 DSN。

### CustdbScripts 类

本节将创建一个名为 CustdbScripts 的 Java 类，其中含有用于处理 ULCustomer upload\_insert 和 download\_cursor 事件的逻辑。可在文本编辑器中输入 CustdbScripts 代码，并将文件保存为 *CustdbScripts.java*。

#### ◆ 创建 CustdbScripts.java:

1. 为 Java 类和程序集创建一个目录。

本教程假定路径为 *c:\mljava*。

2. 使用文本编辑器输入 CustdbScripts 代码:

```
public class CustdbScripts {
    public static String UploadInsert() {
        return("INSERT INTO ULCustomer(cust_id,cust_name) values (?,?)");
    }
    public String DownloadCursor(java.sql.Timestamp ts,String user ) {
        return(
            "SELECT cust_id, cust_name
            FROM ULCustomer where last_modified >= ' " + ts + " ' ");
    }
}
```

**注意：**  
类和关联的方法必须设置为公共的。

3. 在 *c:\mljava* 中将文件保存为 *CustdbScripts.java*。

### 编译 Java 源代码

要执行 Java 同步逻辑，MobiLink 服务器必须能够访问 *mlscript.jar* 中的类。此 jar 文件包含要在 Java 方法中使用的 MobiLink 服务器 API 类的存储库。

为 MobiLink 编译 Java 源代码时，必须包含 *mlscript.jar* 才能使用 MobiLink 服务器 API。本节中使用 javac 实用程序的 `-classpath` 选项来为 CustdbScripts 类指定 *mlscript.jar*。

#### ◆ 编译 Java 源代码 (Windows)

1. 在命令提示符下，导航到包含 *CustdbScripts.java* 的文件夹 (*c:\mljava*)。
2. 键入以下命令。*install-dir* 用 SQL Anywhere 安装所在的位置来替换。

```
javac custdbscripts.java -classpath "install-dir\java\mlscript.jar"
```

将生成 *CustdbScripts.class* 文件。

#### 进一步阅读

有关面向 Java 的 MobiLink 服务器 API 的详细信息，请参见“[用于 Java 的 MobiLink 服务器 API 参考](#)”一节《[MobiLink - 服务器管理](#)》。

有关 Java 方法的详细信息，请参见“[方法](#)”一节《[MobiLink - 服务器管理](#)》。

有关 CustDB 示例数据库和使用备用 RDBMS 服务器的详细信息，请参见“[建立 CustDB 统一数据库](#)”一节第 48 页。

## 第 2 课：指定类方法来处理事件

上一课中创建的 *CustdbScripts.class* 封装 UploadInsert 和 DownloadCursor 方法。这些方法分别包含 ULCustomer upload\_insert 和 download\_cursor 事件的实现。

在本节中，使用两种方法为表级别事件指定类方法：

- ◆ 使用 Sybase Central 中的 [MobiLink 管理] 模式：

使用 Sybase Central 连接到 CustDB 数据库，将 upload\_insert 脚本的语言更改为 Java，然后指定 CustdbScripts.UploadInsert 来处理事件。

- ◆ 使用 ml\_add\_java\_table\_script 存储过程：

使用 Interactive SQL 连接到 CustDB 数据库并执行 ml\_add\_java\_table\_script，指定 CustdbScripts.DownloadCursor 来处理 download\_cursor 事件。

### ◆ 指定 CustdbScripts.UploadInsert 来处理 ULCustomer upload\_insert 事件

1. 使用 Sybase Central 的 [MobiLink 管理] 模式连接到示例数据库：

- ◆ 启动 Sybase Central。
- ◆ 单击 [视图] 菜单，确保已选择 [文件夹]。
- ◆ 在 [连接] 菜单中，选择 [使用 MobiLink 10 连接]。
- ◆ 在 [标识] 选项卡上，选择 ODBC 数据源名称 SQL Anywhere 10 CustDB。
- ◆ 单击 [确定] 进行连接。
- ◆ 现在，Sybase Central 应在 MobiLink 10 插件下显示 CustDB 数据源。

2. 将 ULCustomer 表的现有 upload\_insert 事件删除：

- ◆ 在左窗格中，打开 [同步表] 文件夹，然后选择 ULCustomer 表。在右窗格中，出现一个表级别脚本列表。
- ◆ 单击与 custdb 10.0 upload\_insert 事件关联的表脚本。从 [编辑] 菜单中选择 [删除]。您需要确认方可删除该对象。

3. 为 ULCustomer 表创建一个新的 upload\_insert 事件：

- ◆ 在 [同步表] 文件夹中选择 ULCustomer 表，然后选择 [文件] ► [新建] ► [表脚本]。
- ◆ 选择 upload\_insert 作为要创建的事件，然后单击 [下一步]。
- ◆ 选择使用 Java 语言创建新的脚本定义。
- ◆ 单击 [完成]。

4. 双击 upload\_insert 脚本将其打开。将脚本内容替换为完全限定的方法名 **CustdbScripts.UploadInsert**，然后保存脚本。从 [文件] 菜单中选择 [保存]。

5. 退出 Sybase Central。

此步骤使用 Sybase Central 将 Java 方法指定为 ULCustomer upload\_insert 事件的脚本。

或者，也可以使用 ml\_add\_java\_connection\_script 和 ml\_add\_java\_table\_script 存储过程。使用这些存储过程效率更高，特别是在使用大量 Java 方法处理同步事件时。

请参见“ml\_add\_java\_connection\_script”一节《MobiLink - 服务器管理》和“ml\_add\_java\_table\_script”一节《MobiLink - 服务器管理》。

◆ 指定 CustdbScripts.DownloadCursor() 来处理 ULCustomer download\_cursor 事件

1. 使用 Interactive SQL 连接到示例数据库。

- ◆ 打开 Interactive SQL。

即会出现 [连接] 对话框。

- ◆ 在 [标识] 选项卡上，选择 ODBC 数据源 SQL Anywhere 10 CustDB。
- ◆ 在 [数据库] 选项卡上，确保未选中以下选项：[搜索网络中的数据库服务器]。
- ◆ 单击 [确定] 进行连接。

2. 在 Interactive SQL 中执行以下命令：

```
CALL ml_add_java_table_script(  
  'custdb 10.0',  
  'ULCustomer',  
  'download_cursor',  
  'CustdbScripts.DownloadCursor');  
COMMIT;
```

以下是对每个参数的说明：

参数	说明
custdb 10.0	脚本版本。
ULCustomer	同步表。
download_cursor	事件名称。
CustdbScripts.DownloadCursor	完全限定的 Java 方法。

3. 退出 Interactive SQL。

在本课中，您指定了 Java 方法来处理 ULCustomer 表级别事件。下一课将确保 MobiLink 服务器装载合适的类文件和 MobiLink 服务器 API。

**进一步阅读**

有关使用存储过程来添加脚本的详细信息，请参见“ml\_add\_java\_connection\_script”一节《MobiLink - 服务器管理》和“ml\_add\_java\_table\_script”一节《MobiLink - 服务器管理》。

有关添加和删除同步脚本的一般信息，请参见“添加和删除脚本”一节《MobiLink - 服务器管理》。

## 第 3 课：用 -sl java 运行 MobiLink 服务器

用 -sl java -cp 选项运行 MobiLink 服务器可以指定一组目录来搜索其中的类文件，并强制在服务器启动时装载 Java 虚拟机。

### ◆ 启动 MobiLink 服务器 (mlsrv10) 并装载 Java 程序集

- 在命令提示符处，在一行内键入以下命令：

```
mlsrv10 -c "dsn=SQL Anywhere 10 CustDB" -sl java (-cp c:\mljava)
```

显示一条消息指出服务器已启动。现在，Java 方法会在同步过程中 ULCustomer 表的 upload\_insert 事件被触发时执行。

### 进一步阅读

有关详细信息，请参见“-sl java 选项”一节《MobiLink - 服务器管理》。

## 第 4 课：测试同步

UltraLite 随附一个示例 Windows 客户端，当用户发出同步请求时，该客户端自动调用 dbmlsync 实用程序。它是一个简单的销售状态应用程序，您可以对上一课中启动的 CustDB 统一数据库运行该程序。

### 启动应用程序 (Windows)

#### ◆ 启动并同步示例应用程序

1. 启动示例应用程序。  
从 [开始] 菜单中，选择 [程序] ► [SQL Anywhere 10] ► [UltraLite] ► [Windows 示例应用程序]。
2. 输入一个职员 ID。  
键入值 **50** 并按 Enter 键。

应用程序会自动同步，同时会将一组客户、产品和订单从 CustDB 统一数据库下载到应用程序中。

### 添加订单 (Windows)

#### ◆ 添加订单：

1. 从 [Order] 菜单中，选择 [New]。  
即会出现 [Add New Order] 屏幕。
2. 输入新的客户名称。  
例如，输入 **Frank Javac**。
3. 选择一种产品，然后输入数量和折扣。
4. 按回车键以添加新订单。

您现在已经修改了本地 UltraLite 数据库中的数据。在进行同步之前，此数据没有与统一数据库共享。

#### ◆ 与统一数据库进行同步并触发 upload\_insert 事件

- 从 [File] 菜单中，选择 [Synchronize]。

出现一个窗口，显示已将一个插入操作成功上载到统一数据库中。

### 进一步阅读

有关 CustDB Windows 应用程序的详细信息，请参见“[探讨 MobiLink 的 CustDB 示例](#)”第 45 页。

## 清除

从计算机中删除教程资料。

### ◆ 从计算机中删除教程资料

1. 将 ULCustomer 表的 upload\_insert 和 download\_cursor 脚本返回到其原始 SQL 逻辑。

- ◆ 打开 Interactive SQL。

即会出现 [连接] 对话框。

- ◆ 在 [标识] 选项卡上，选择 ODBC 数据源 SQL Anywhere 10 CustDB。
- ◆ 单击 [确定] 进行连接。
- ◆ 在 Interactive SQL 中执行以下命令：

```
CALL ml_add_table_script( 'custdb 10.0',  
    'ULCustomer',  
    'upload_insert',  
    'INSERT INTO ULCustomer( cust_id, cust_name ) VALUES( ?, ? )' );  
  
CALL ml_add_table_script( 'custdb 10.0',  
    'ULCustomer',  
    'download_cursor',  
    'SELECT "cust_id", "cust_name"  
    FROM "ULCustomer" WHERE "last_modified" >= ?' );  
COMMIT;
```

2. 关闭 SQL Anywhere 窗口、MobiLink 窗口和同步客户端窗口，方法是右击各个任务栏项并选择 [关闭]。

3. 删除所有与教程相关的 Java 文件。

删除包含您的 *CustdbScripts.java* 和 *CustdbScripts.class* 文件的文件夹 (*c:\mljava*)。

**注意：**

确保 *c:\mljava* 中没有其它重要文件。

---

## 进一步阅读

有关使用 Java 编写 MobiLink 同步脚本的详细信息，请参见“[设置 Java 同步逻辑](#)”一节《[MobiLink - 服务器管理](#)》。

有关说明自定义验证的 Java 同步脚本用法的示例，请参见“[Java 同步示例](#)”一节《[MobiLink - 服务器管理](#)》。

有关同步脚本编写的详细信息，请参见“[编写同步脚本](#)”《[MobiLink - 服务器管理](#)》和“[同步事件](#)”《[MobiLink - 服务器管理](#)》。

有关其它同步方法（如时间戳）的介绍，请参见“[同步技术](#)”《[MobiLink - 服务器管理](#)》。

---

---

## 第 7 章

# 教程：使用 .NET 同步逻辑

## 目录

.NET 同步教程简介 .....	104
第 1 课：用 MobiLink 参考编译 CustdbScripts.dll 程序集 .....	105
第 2 课：为事件指定类方法 .....	109
第 3 课：带 -sl dnet 运行 MobiLink .....	112
第 4 课：测试同步 .....	113
清除 .....	115
进一步阅读 .....	116

## .NET 同步教程简介

本教程将指导您完成使用 .NET 同步逻辑的基本步骤。通过使用 CustDB 示例作为 SQL Anywhere 统一数据库，可以为 MobiLink 表级事件指定简单的类方法。该过程还涉及带选项运行 MobiLink 服务器 (mlsrv10)，所带选项用于设置 .NET 程序集的路径。

### 必需的软件

- ◆ SQL Anywhere 10.0
- ◆ Microsoft .NET Framework SDK

### 能力和经验

您应该：

- ◆ 熟悉 .NET
- ◆ 了解 MobiLink 事件脚本的基本知识

### 目标

您将掌握并熟悉以下内容：

- ◆ 利用 .NET 类方法编写 MobiLink 表级事件脚本

### 重要概念

本节通过以下步骤使用 MobiLink CustDB 示例数据库实现基本 .NET 同步：

- ◆ 用 MobiLink 参考编译 *CustdbScripts.dll* 专用程序集
- ◆ 为表级事件指定类方法
- ◆ 带 -sl dnet 选项运行 MobiLink 服务器 (mlsrv10)
- ◆ 使用示例 Windows 客户端应用程序测试同步

### 建议阅读的背景知识

有关同步脚本的详细信息，请参见“同步脚本介绍”一节 [《MobiLink - 服务器管理》](#)。

## 第 1 课：用 MobiLink 参考编译 CustdbScripts.dll 程序集

.NET 类将同步逻辑封装在方法中。

在本课中，您将编译一个与 CustDB 示例数据库关联的类。

### MobiLink 数据库示例

SQL Anywhere 随附一个 SQL Anywhere 示例数据库 (CustDB)，该数据库已设置为可以进行同步，其中包括驱动同步所需的 SQL 脚本。例如，CustDB ULCustomer 表是一个支持多种表级事件的同步表。

CustDB 的设计用途是作为 UltraLite 和 SQL Anywhere 客户端的统一数据库服务器。CustDB 数据库有一个名为 SQL Anywhere 10 CustDB 的 DSN。

### CustdbScripts 程序集

在本节中，您将创建一个名为 CustdbScripts 的 .NET 类，其中包含用于处理 ULCustomer upload\_insert 和 download\_cursor 事件的逻辑。

### MobiLink 服务器 API

要执行 .NET 同步逻辑，MobiLink 服务器必须能够访问 *iAnywhere.MobiLink.Script.dll* 中的类。*iAnywhere.MobiLink.Script.dll* 包含要在 .NET 方法中使用的面向 .NET 的 MobiLink 服务器 API 类的存储库。

有关用于 .NET 的 MobiLink 服务器 API 的详细信息，请参见“用于 .NET 参考的 MobiLink 服务器 API”一节《MobiLink - 服务器管理》。

编译 CustdbScripts 类时，必须包含此程序集才能使用该 API。可以使用 Visual Studio .NET 或在命令提示符中编译类。

- ◆ 在 Visual Studio .NET 中，创建一个新的类库，并输入 CustdbScripts 代码。链接 *iAnywhere.MobiLink.Script.dll*，并构建类的程序集。
- ◆ 在命令提示符中，使用文本编辑器输入 CustdbScripts 代码并将文件保存为 *CustdbScripts.cs*（如果使用 Visual Basic .NET，则保存为 *CustdbScripts.vb*）。使用命令行编译器，引用 *iAnywhere.MobiLink.Script.dll* 并构建类的程序集。

#### ◆ 使用 Visual Studio .NET 创建 CustdbScripts 程序集

1. 启动一个新的 Visual C# 或 Visual Basic .NET 类库项目。  
使用 CustdbScripts 作为项目名，并输入相应的路径。本教程假定路径为 *c:\mldnet*。
2. 输入 CustdbScripts 代码。  
如果是 C#，请键入：

```
namespace MLEExample
{
    class CustdbScripts
    {
```

```
        public static string UploadInsert()
        {
            return("INSERT INTO ULCustomer(cust_id,cust_name) values (?,?)");
        }
        public static string DownloadCursor(System.DateTime ts, string user )
        {
            return("SELECT cust_id, cust_name
                FROM ULCustomer
                WHERE last_modified >= '" + ts.ToString("yyyy-MM-dd
                hh:mm:ss.fff") + "'");
        }
    }
}
```

如果是 Visual Basic .NET，请键入：

```
Namespace MLExample

    Class CustdbScripts

        Public Shared Function UploadInsert() As String
            Return("INSERT INTO ULCustomer(cust_id,cust_name) values (?,?)")
        End Function

        Public Shared Function DownloadCursor(ByVal ts As System.DateTime,
        ByVal user As String) As String
            Return("SELECT cust_id, cust_name FROM ULCustomer " +
                "WHERE last_modified >= '" + ts.ToString("yyyy-MM-dd hh:mm:ss.fff")
                + "'")
        End Function

    End Class

End Namespace
```

3. 添加对 MobiLink 服务器 API 的引用。

- ◆ 从 Visual Studio .NET 的 [Project] 菜单中选择 [Add Existing Item...]
- ◆ 在 SQL Anywhere 安装目录的 *Assembly\vl* 子目录中，选择 *iAnywhere.MobiLink.Script.dll*。在 Visual Studio .NET 的 [Open] 下拉菜单中，选择 [Link File]。在 Visual Studio 2005 的 [Add] 菜单中选择 [Add Link]。

4. 右击 CustdbScripts 项目并选择表 [Common Properties] ► [General]。确保称为 Root Namespace 的文本框中没有任何文本。

5. 生成 *CustdbScripts.dll*。

在 [Build] 菜单中，选择 [Build CustdbScripts]。

这会在 *C:\mldnet\CustdbScripts\CustdbScripts\bin\Debug* 中创建 *CustdbScripts.dll*。

#### ◆ 在命令提示符中创建 CustdbScripts 程序集

1. 为 .NET 类和程序集创建一个目录。

本教程假定路径为 *c:\mldnet*。

2. 使用文本编辑器输入 CustdbScripts 代码。

如果是 C#，请键入：

```
namespace MLExample
{
class CustdbScripts
{
    public static string UploadInsert()
    {
        return("INSERT INTO ulcustomer(cust_id,cust_name) values (?,?)");
    }
    public static string DownloadCursor(System.DateTime ts, string user )
    {
        return("SELECT cust_id, cust_name FROM ULCustomer where last_modified
>= '" + ts.ToString("yyyy-MM-dd hh:mm:ss.fff") + "'");
    }
}
}
```

如果是 Visual Basic .NET，请键入：

```
Namespace MLExample

Class CustdbScripts

    Public Shared Function UploadInsert() As String
        Return("INSERT INTO ULCustomer(cust_id,cust_name) values (?,?)")
    End Function

    Public Shared Function DownloadCursor(ByVal ts As System.DateTime,
ByVal user As String) As String
        Return("SELECT cust_id, cust_name FROM ULCustomer " +
"WHERE last_modified >= '" + ts.ToString("yyyy-MM-dd hh:mm:ss.fff")
+ "'")
    End Function

End Class

End Namespace
```

3. 在 `c:\mldnet` 中将文件保存为 `CustdbScripts.cs`（如果使用 Visual Basic .NET，则保存为 `CustdbScripts.vb`）。
4. 使用以下命令编译该文件。

如果是 C#，请键入：

```
csc /out:c:\mldnet\custdbscripts.dll /target:library /reference:"%
sqlany10%\Assembly\v1\iAnywhere.MobiLink.Script.dll" c:\mldnet
\CustdbScripts.cs
```

如果是 Visual Basic .NET，请键入：

```
vbc /out:c:\mldnet\custdbscripts.dll /target:library /reference:"%
sqlany10%\Assembly\v1\iAnywhere.MobiLink.Script.dll" c:\mldnet
\CustdbScripts.vb
```

将生成 `CustdbScripts.dll` 程序集。

### 进一步阅读

有关用于 .NET 的 MobiLink 服务器 API 的详细信息，请参见“用于 .NET 参考的 MobiLink 服务器 API”一节《MobiLink - 服务器管理》。

有关 .NET 方法的详细信息，请参见“方法”一节 [《MobiLink - 服务器管理》](#)。

## 第 2 课：为事件指定类方法

有关 CustDB 示例数据库的详细信息，请参见“[建立 CustDB 统一数据库](#)”一节第 48 页。

上一课中创建的 *CustdbScripts.dll* 封装了 UploadInsert() 和 DownloadCursor() 方法。这些方法分别包含 ULCustomer upload\_insert 和 download\_cursor 事件的实现。

在本节中，可以用两种方法为表级事件指定类方法：

1. 使用 MobiLink 同步插件。

使用 Sybase Central 连接到 CustDB 数据库，将 upload\_insert 脚本的语言更改为 .NET，然后指定 MLExample.CustdbScripts.UploadInsert 来处理事件。

2. 使用 ml\_add\_dnet\_table\_script 存储过程。

您将通过 Interactive SQL 连接到 CustDB 数据库并执行 ml\_add\_dnet\_table\_script，从而为 download\_cursor 事件指定 MLExample.CustdbScripts.DownloadCursor。

### ◆ 为 CustdbScripts.uploadInsert() 预订 ULCustomer 表的 upload\_insert 事件

1. 使用 MobiLink 同步插件连接到示例数据库：

- ◆ 启动 Sybase Central。
- ◆ 在 [视图] 菜单中，确保已选择 [文件夹]。
- ◆ 在 [连接] 菜单中，选择 [使用 MobiLink 10 连接]。
- ◆ 在 [标识] 选项卡上，选择 ODBC 数据源名 SQL Anywhere 10 CustDB。
- ◆ 单击 [确定] 进行连接。
- ◆ 现在，Sybase Central 应在 MobiLink 10 插件下显示 CustDB 数据源。

2. 将 ULCustomer 表 upload\_insert 事件的语言更改为 .NET：

- ◆ 在左窗格中，打开 [同步表] 文件夹，然后选择 ULCustomer 表。右窗格中将出现表级脚本的列表。
- ◆ 单击与 custdb 10.0 upload\_insert 事件关联的表脚本。在 [文件] 菜单中，选择 [语言]，并将语言更改为 .NET。

3. 输入完全限定 .NET 方法名作为 upload\_insert 脚本。

- ◆ 双击与 upload\_insert 事件关联的表脚本。  
将出现一个窗口，其中显示了脚本内容。
- ◆ 将脚本内容更改为完全限定方法名 MLExample.CustdbScripts.UploadInsert。

**注意：**  
完全限定方法名区分大小写。

- ◆ 要保存脚本，请从 [文件] 菜单中选择 [保存]。

4. 退出 Sybase Central。

此步骤使用 Sybase Central 将 .NET 方法指定为 ULCustomer upload\_insert 事件的脚本。

也可以使用 ml\_add\_dnet\_connection\_script 和 ml\_add\_dnet\_table\_script 存储过程。使用这些存储过程效率更高，特别是在使用大量 .NET 方法处理同步事件时。

请参见“ml\_add\_dnet\_connection\_script”一节《MobiLink - 服务器管理》和“ml\_add\_dnet\_table\_script”一节《MobiLink - 服务器管理》。

在下一节中，您将通过 Interactive SQL 连接到 CustDB 并执行 ml\_add\_dnet\_table\_script，从而将 MLEExample.CustdbScripts.DownloadCursor 指派给 download\_cursor 事件。

◆ 为 ULCustomer download\_cursor 事件指定 MLEExample.CustdbScripts.DownloadCursor

1. 使用 Interactive SQL 连接到示例数据库。

- ◆ 启动 Interactive SQL：

选择 [开始] ► [程序] ► [SQL Anywhere 10] ► [Interactive SQL]，或在命令提示符中键入以下命令：

```
dbisql
```

将出现 [连接] 对话框。

- ◆ 在 [标识] 选项卡上，选择 ODBC 数据源 SQL Anywhere 10 CustDB。
- ◆ 在 [数据库] 选项卡上，确保未选中用于搜索网络数据库服务器的选项。
- ◆ 单击 [确定] 进行连接。

2. 在 Interactive SQL 中执行以下命令：

```
CALL ml_add_dnet_table_script(  
  'custdb 10.0',  
  'ULCustomer',  
  'download_cursor',  
  'MLEExample.CustdbScripts.DownloadCursor');  
COMMIT;
```

以下是对每个参数的说明：

参数	说明
custdb 10.0	脚本版本。
ULCustomer	同步表。

参数	说明
download_cursor	事件名。
MLExample.CustdbScripts.DownloadCursor	完全限定 .NET 方法。

### 3. 退出 Interactive SQL。

在本课中，您指定了 .NET 方法来处理 ULCustomer 表级事件。下一课将确保 MobiLink 服务器装载正确的类文件和 MobiLink 服务器 API。

### 进一步阅读

有关添加和删除同步脚本的详细信息，请参见“[添加和删除脚本](#)”一节《[MobiLink - 服务器管理](#)》。

有关本课所用脚本的详细信息，请参见“[ml\\_add\\_dnet\\_connection\\_script](#)”一节《[MobiLink - 服务器管理](#)》和“[ml\\_add\\_dnet\\_table\\_script](#)”一节《[MobiLink - 服务器管理](#)》。

## 第 3 课：带 -sl dnet 运行 MobiLink

带 -sl dnet 选项运行 MobiLink 服务器可以指定 .NET 程序集的位置，并强制在服务器启动时装载 CLR。

如果使用 Visual Studio .NET 进行编译，则 *CustdbScripts.dll* 的位置是 *c:\mldnet\CustdbScripts\CustdbScripts\bin\Debug*。如果在命令提示符中进行编译，则 *CustdbScripts.dll* 的位置是 *c:\mldnet*。

### ◆ 启动 MobiLink 服务器 (mlsrv10) 并装载 .NET 程序集

- 带 -sl dnet 选项启动 MobiLink 服务器。

如果使用 Visual Studio .NET 编译程序集：

在命令提示符中，在一行上键入以下命令：

```
mlsrv10 -c "dsn=SQL Anywhere 10 CustDB" -dl -o cons1.txt -v+ -sl dnet(-MLAutoLoadPath=c:\mldnet\CustdbScripts\CustdbScripts\bin\Debug)
```

如果在命令提示符中编译程序集：

在命令提示符处，在一行上键入以下命令：

```
mlsrv10 -c "dsn=SQL Anywhere 10 CustDB" -dl -o cons1.txt -v+ -sl dnet(-MLAutoLoadPath=c:\mldnet)
```

将出现一个消息对话框，指示服务器已做好了处理请求的准备。现在如果同步过程中触发了 `upload_insert` 事件，便会执行 .NET 方法。

### 进一步阅读

有关详细信息，请参见“[-sl dnet 选项](#)”一节《[MobiLink - 服务器管理](#)》。

## 第 4 课：测试同步

UltraLite 随附一个示例 Windows 客户端，当用户发出同步请求时，该客户端自动调用 dbmlsync 实用程序。它是一个简单的销售状态应用程序，您可以对上一课中启动的 CustDB 统一数据库运行该应用程序。

### 启动应用程序

#### ◆ 启动并同步示例应用程序

1. 启动示例应用程序。

从 [开始] 菜单中，选择 [程序] ► [SQL Anywhere 10] ► [UltraLite] ► [Windows 示例应用程序]。

2. 输入一个职员 ID。

输入值 **50** 并按 Enter 键。

应用程序会自动进行同步，将一组客户、产品和订单从 CustDB 统一数据库下载到应用程序中。

在下一节中，您将输入新的客户名和订单详细信息。在随后进行的同步期间，这些信息将会上载到 CustDB 统一数据库中，并会触发 ULCustomer 表的 upload\_insert 和 download\_cursor 事件。

### 添加订单

#### ◆ 添加订单：

1. 从 [Order] 菜单中，选择 [New]。  
将出现 [Add New Order] 对话框。
2. 输入新的 [Customer Name]。  
例如，输入 [Frank DotNET]。
3. 选择一种产品，然后输入数量和折扣。
4. 按 Enter 键添加新订单。

您现在已经修改了本地 UltraLite 数据库中的数据。进行同步时这些数据才会与统一数据库共享。

#### ◆ 与统一数据库进行同步并触发 upload\_insert 事件

- 从 [文件] 菜单中选择 [同步]。  
将出现一个窗口，指出已将一个插入操作成功上载到统一数据库中。

### 进一步阅读

有关 CustDB Windows 应用程序的详细信息，请参见“[探讨 MobiLink 的 CustDB 示例](#)”第 45 页。

## 清除

从计算机中删除教程资料。

### ◆ 从计算机中删除教程资料

1. 将 ULCustomer 表的 upload\_insert 和 download\_cursor 脚本恢复为其原始 SQL 逻辑。

- ◆ 打开 Interactive SQL。

选择 [开始] ► [程序] ► [SQL Anywhere 10] ► [Interactive SQL]，或在命令提示符中键入以下命令：

```
dbisql
```

将出现 [连接] 对话框。

- ◆ 在 [标识] 选项卡上，选择 ODBC 数据源 SQL Anywhere 10 CustDB。
- ◆ 单击 [确定] 进行连接。
- ◆ 在 Interactive SQL 中执行以下命令：

```
CALL ml_add_table_script( 'custdb 10.0',  
    'ULCustomer',  
    'upload_insert',  
    'INSERT INTO ULCustomer( cust_id, cust_name ) VALUES( ?, ? )' );  
  
CALL ml_add_table_script( 'custdb 10.0',  
    'ULCustomer',  
    'download_cursor',  
    'SELECT "cust_id", "cust_name"  
    FROM "ULCustomer"  
    WHERE "last_modified" >= ?' );
```

2. 右击 SQL Anywhere 窗口、MobiLink 窗口和同步客户端窗口并选择 [关闭] 来将它们关闭。
3. 删除所有与教程有关的 .NET 源文件。

删除包含您的 *CustdbScripts.cs* 和 *CustdbScripts.dll* 文件的文件夹 (*c:\mldnet*)。

**注意：**

确保 *c:\mldnet* 中没有其它重要文件。

## 进一步阅读

有关使用 .NET 编写 MobiLink 同步脚本的详细信息，请参见“[设置 .NET 同步逻辑](#)”一节《[MobiLink - 服务器管理](#)》。

有关调试 .NET 同步逻辑的信息，请参见“[调试 .NET 同步逻辑](#)”一节《[MobiLink - 服务器管理](#)》。

有关说明使用 .NET 同步脚本进行自定义验证的详细示例，请参见“[.NET 同步示例](#)”一节《[MobiLink - 服务器管理](#)》。

有关同步脚本编写的详细信息，请参见“[编写同步脚本](#)”《[MobiLink - 服务器管理](#)》和“[同步事件](#)”《[MobiLink - 服务器管理](#)》。

有关其它同步方法（如时间戳）的介绍，请参见“[同步技术](#)”《[MobiLink - 服务器管理](#)》。

---

## 第 8 章

# 教程：使用 .NET 和 Java 进行自定义验证

## 目录

MobiLink 自定义验证简介 .....	118
第 1 课：创建用于自定义验证的 Java 或 .NET 类（服务器端） .....	119
第 2 课：为 authenticate_user 事件注册 Java 或 .NET 脚本 .....	122
第 3 课：启动面向 Java 或 .NET 的 MobiLink 服务器 .....	123
第 4 课：测试验证 .....	124
清除 .....	125
进一步阅读 .....	126

## MobiLink 自定义验证简介

您可以使用 SQL、Java 或 .NET 编写 MobiLink 同步脚本。您可以使用 Java 或 .NET 在同步的任意时刻添加自定义操作。

在本教程内，您将为 `authenticate_user` 连接事件添加 Java 或 .NET 方法。`authenticate_user` 事件允许您指定一个自定义验证方案并替换 MobiLink 内置客户端验证。

### 必需的软件

- ◆ SQL Anywhere 10.0
- ◆ Java 软件开发工具包

### 能力和经验

您需要：

- ◆ 熟悉 Java
- ◆ 了解 MobiLink 事件脚本的基本知识

### 目标

您将掌握并熟悉以下内容：

- ◆ MobiLink 自定义验证

### 重要概念

本节使用以下步骤通过 MobiLink CustDB 示例数据库实现基本的基于 Java 的同步：

- ◆ 使用 MobiLink 服务器 API 引用编译源文件
- ◆ 为特定表级别事件指定类方法
- ◆ 用 `-sl java` 选项运行 MobiLink 服务器 (`mlsrv10`)
- ◆ 使用示例 Windows 客户端应用程序测试同步

### 建议阅读的背景知识

有关验证 MobiLink 客户端的详细信息，请参见“[选择用户验证机制](#)”一节《[MobiLink - 客户端管理](#)》。

有关集成 POP3、IMAP 或 LDAP 验证的详细信息，请参见“[向外部服务器验证](#)”一节《[MobiLink - 客户端管理](#)》。

有关 .NET 或 Java 同步脚本的详细信息，请参见“[使用 .NET 编写同步脚本](#)”《[MobiLink - 服务器管理](#)》或“[使用 Java 语言编写同步脚本](#)”《[MobiLink - 服务器管理](#)》。

## 第 1 课：创建用于自定义验证的 Java 或 .NET 类（服务器端）

在本课中，您将编译用于自定义验证的包含 Java 或 .NET 逻辑的类。

### 面向 .NET 的 MobiLink 服务器 API

要执行 .NET 同步逻辑，MobiLink 服务器必须能够访问 *iAnywhere.MobiLink.Script.dll* 中的类。*iAnywhere.MobiLink.Script.dll* 包含要在 .NET 方法中使用的 MobiLink .NET 服务器 API 类的存储库。编译 .NET 类时，您会引用 *iAnywhere.MobiLink.Script.dll*。

### 面向 Java 的 MobiLink 服务器 API

要执行 Java 同步逻辑，MobiLink 服务器必须能够访问 *mlscript.jar* 中的类。*mlscript.jar* 包含要在 Java 方法中使用的 MobiLink Java 服务器 API 类的存储库。编译 Java 类时，您会引用 *mlscript.jar*。

#### ◆ 创建用于自定义验证的 Java 或 .NET 类

1. 使用 Java 或 .NET 创建名为 MobiLinkAuth 的类。

MobiLinkAuth 类包括用于 `authenticate_user` 同步事件的 `authenticateUser` 方法。`authenticate_user` 事件可为用户和口令提供参数。在 `authentication_status` inout 参数中返回验证结果。

对于 Java，键入以下语句：

```
import ianywhere.ml.script.*;

public class MobiLinkAuth
{

    public void authenticateUser (
        ianywhere.ml.script.InOutInteger authentication_status,
        String user,
        String pwd,
        String newPwd )
    {
        // to do...
    }

}
```

对于 .NET，键入以下语句：

```
using iAnywhere.MobiLink.Script;

public class MobiLinkAuth
{
    public void authenticateUser (
        ref int authentication_status,
        string user,
        string pwd,
        string newPwd)
    {

        // to do...
    }
}
```

```
    }  
}
```

## 2. 编写 authenticateUser 方法。

本教程说明一个非常简单的自定义用户验证的情况。如果用户名以 "ML" 开头，则验证成功。

### 注意：

您将在“[第 2 课：为 authenticate\\_user 事件注册 Java 或 .NET 脚本](#)”一节第 122 页内注册 authenticate\_user 同步事件的 authenticateUser 方法。

对于 Java，键入以下语句：

```
public void authenticateUser (  
    anywhere.ml.script.InOutInteger authentication_status,  
    String user,  
    String pwd,  
    String newPwd ) {  
  
    if(user.substring(0,1)== "ML") {  
        // success: an auth status code of 1000  
        authentication_status.setValue(1000);  
    } else {  
        // fail: an authentication_status code of 4000  
        authentication_status.setValue(4000);  
    }  
}
```

对于 .NET，键入以下语句：

```
public void authenticateUser(  
    ref int authentication_status,  
    string user,  
    string pwd,  
    string newPwd ) {  
  
    if(user.Substring(0,2)== "ML") {  
        // success: an auth status code of 1000  
        authentication_status = 1000;  
    } else {  
        // fail: and authentication_status code of 4000  
        authentication_status = 4000;  
    }  
}
```

## 3. 编译 MobiLinkAuth 类。

- ◆ 对于 .NET，将文件另存为 *MobiLinkAuth.cs*（位于 *c:\mlauth* 下）。
- ◆ 在命令提示符下，转到 *c:\mlauth*。在一行内键入以下命令来编译文件。*install-dir* 用 SQL Anywhere 安装所在的位置来替换。

```
csc /out:c:\mlauth\MobiLinkAuth.dll /target:library  
/reference:install-dir\Assembly\v1\iAnywhere.MobiLink.Script.dll"  
MobiLinkAuth.cs
```

将生成 *MobiLinkAuth.dll* 程序集。

- ◆ 对于 Java，将文件另存为 *MobiLinkAuth.java*（位于 *c:\mlauth* 下）。在命令提示符下，转到 *c:\mlauth*。要编译文件，请在一行内键入以下命令。*install-dir* 用 SQL Anywhere 安装所在的位置来替换。

```
javac MobiLinkAuth.java -classpath "install-dir/java/mlscript.jar"
```

将生成 *MobiLinkAuth.class* 文件。

### 进一步阅读

有关 `authenticate_user` 事件（包括 `authentication_status` 返回代码表）的详细信息，请参见“[authenticate\\_user 连接事件](#)”一节《[MobiLink - 服务器管理](#)》。

有关使用 Java 或 .NET 执行自定义验证的详细信息，请参见“[Java 和 .NET 用户验证](#)”一节《[MobiLink - 客户端管理](#)》。

有关 Java 自定义验证的详细示例，请参见“[Java 同步示例](#)”一节《[MobiLink - 服务器管理](#)》。

有关 .NET 自定义验证的详细示例，请参见“[.NET 同步示例](#)”一节《[MobiLink - 服务器管理](#)》。

## 第 2 课：为 authenticate\_user 事件注册 Java 或 .NET 脚本

在本课中，您将为 authenticate\_user 同步事件注册 MobiLinkAuth authenticateUser 方法。您将把此脚本添加到 MobiLink 的示例数据库 CustDB 中。

### MobiLink 数据库示例

SQL Anywhere 随附一个已针对同步进行了设置的 SQL Anywhere 示例数据库 (CustDB)。举例来说，CustDB ULCustomer 表是一个同步表，它支持多种表级别脚本。

CustDB 的设计用途是作为 UltraLite 和 SQL Anywhere 客户端的统一数据库服务器。CustDB 数据库有一个名为 SQL Anywhere 10 CustDB 的 DSN。

#### ◆ 注册 authenticate\_user 事件的 authenticateUser 方法

1. 使用 Interactive SQL 连接到示例数据库。

在命令提示符处键入：

```
dbisql -c "dsn=SQL Anywhere 10 CustDB"
```

2. 使用先前存储的 ml\_add\_java\_connection\_script 或 ml\_add\_dnet\_connection\_script 注册 authenticate\_user 事件的 authenticateUser 方法。

对于 Java，在 Interactive SQL 中执行以下命令：

```
call ml_add_java_connection_script(  
'custdb 10.0',  
'authenticate_user',  
'MobiLinkAuth.authenticateUser');  
commit;
```

对于 .NET，在 Interactive SQL 中执行以下命令：

```
call ml_add_dnet_connection_script(  
'custdb 10.0',  
'authenticate_user',  
'MobiLinkAuth.authenticateUser');  
commit;
```

在下一课中，您将启动 MobiLink 服务器并加载您的类文件或程序集。

### 进一步阅读

有关添加和删除同步脚本的一般信息，请参见“[添加和删除脚本](#)”一节《[MobiLink - 服务器管理](#)》。

有关 ml\_add\_java\_connection\_script 的详细信息，请参见“[ml\\_add\\_java\\_connection\\_script](#)”一节《[MobiLink - 服务器管理](#)》。

有关 ml\_add\_dnet\_connection\_script 的详细信息，请参见“[ml\\_add\\_dnet\\_connection\\_script](#)”一节《[MobiLink - 服务器管理](#)》。

## 第 3 课: 启动面向 Java 或 .NET 的 MobiLink 服务器

以 `-sl java` 或 `-sl dnet` 选项启动 MobiLink 服务器可允许您指定一组目录来搜索已编译的文件。

### ◆ 启动 MobiLink 服务器 (mlsrv10)

- 连接到 MobiLink 示例数据库, 然后在 `mlsrv10` 命令行装载 Java 类或 .NET 程序集。

对于 Java, 在命令提示符下键入以下命令:

```
mlsrv10 -c "dsn=SQL Anywhere 10 CustDB" -sl java(-cp c:\mlauth)
```

对于 .NET, 在命令提示符下键入以下命令:

```
mlsrv10 -c "dsn=SQL Anywhere 10 CustDB" -sl dnet(-MLAutoLoadPath=c:\mlauth)
```

即会出现 MobiLink 服务器窗口。现在, 当 `authenticate_user` 同步事件出现时, 您的 Java 或 .NET 方法将会执行。

### 进一步阅读

有关启动面向 Java 的 MobiLink 服务器的详细信息, 请参见 [“-sl java 选项”](#) 一节 《MobiLink - 服务器管理》。

有关启动面向 .NET 的 MobiLink 服务器的详细信息, 请参见 [“-sl dnet 选项”](#) 一节 《MobiLink - 服务器管理》。

## 第 4 课：测试验证

UltraLite 随附一个示例 Windows 客户端，当用户发出同步请求时，该客户端自动调用 dbmlsync 实用程序。它是一个简单的销售状态应用程序，您可以对上一课中启动的 CustDB 统一数据库运行该程序。

### ◆ 启动示例应用程序并测试验证

1. 启动示例应用程序。

从 [开始] 菜单中，选择 [程序] ► [SQL Anywhere 10] ► [UltraLite] ► [Windows 示例应用程序]。

2. 输入一个无效的雇员 ID 并进行同步。

在此应用程序中，雇员 ID 也是 MobiLink 用户名。如果用户名不是以 "ML" 开头，则 Java 或 .NET 逻辑将使得同步失败。为雇员 ID 输入值 **50** 并按回车键。

即会出现 UltraLite CustDB Demo 对话框，该对话框使用 SQL 代码 -103 表示同步错误。SQL 代码 -103 代表无效的用户 ID 或口令。

### 进一步阅读

有关 CustDB Windows 应用程序的详细信息，请参见 [“探讨 MobiLink 的 CustDB 示例” 第 45 页](#)。

## 清除

从计算机中删除教程资料。

### ◆ 从计算机中删除教程资料

1. 从统一数据库中删除 `authenticate_user` 脚本。

- ◆ 使用 Interactive SQL 连接到 MobiLink 示例数据库。

在命令提示符处键入：

```
dbisql -c "dsn=SQL Anywhere 10 CustDB"
```

- ◆ 删除 `authenticate_user` 脚本。

对于 Java，执行下列命令来删除 `authenticate_user` 脚本：

```
call ml_add_java_connection_script(  
'custdb_10.0',  
'authenticate_user',  
null);  
commit;
```

对于 .NET，执行下列命令来删除 `authenticate_user` 脚本：

```
call ml_add_dnet_connection_script(  
'custdb_10.0',  
'authenticate_user',  
null);  
commit;
```

2. 删除 Java 或 .NET 源文件。

例如，删除 `c:\mlauth` 目录。

#### 小心

确保在此目录中只有与教程相关的资料。

3. 关闭 Interactive SQL 和 UltraLite Windows 客户端应用程序。

从每个应用程序的 [文件] 菜单中选择 [退出]。

4. 关闭 SQL Anywhere、MobiLink 和同步客户端窗口。

右击每个任务栏项并选择 [关闭]。

## 进一步阅读

有关 Java 同步逻辑的详细信息，请参见“[使用 Java 语言编写同步脚本](#)”《[MobiLink - 服务器管理](#)》。

有关 .NET 同步逻辑的详细信息，请参见“[使用 .NET 编写同步脚本](#)”《[MobiLink - 服务器管理](#)》。

有关说明自定义验证的 Java 或 .NET 同步脚本用法的详细示例，请分别参见“[Java 同步示例](#)”一节《[MobiLink - 服务器管理](#)》或“[.NET 同步示例](#)”一节《[MobiLink - 服务器管理](#)》。

有关调试 Java 或 .NET 同步逻辑的信息，请分别参见“[调试 Java 类](#)”一节《[MobiLink - 服务器管理](#)》或“[调试 .NET 同步逻辑](#)”一节《[MobiLink - 服务器管理](#)》。

有关同步脚本的详细信息，请参见“[编写同步脚本](#)”《[MobiLink - 服务器管理](#)》和“[同步事件](#)”《[MobiLink - 服务器管理](#)》。

---

## 第 9 章

# 教程：使用直接行处理

## 目录

直接行处理教程简介 .....	128
第 1 课：建立 MobiLink 统一数据库 .....	129
第 2 课：添加同步脚本 .....	132
第 3 课：为处理直接行处理编写 Java 或 .NET 逻辑 .....	135
第 4 课：启动 MobiLink 服务器 .....	146
第 5 课：建立 MobiLink 客户端 .....	147
第 6 课：同步 .....	149
清除 .....	151
进一步阅读 .....	152

## 直接行处理教程简介

此教程介绍如何实现 MobiLink 直接行处理，以便您可使用支持的统一数据源以外的其它数据源。

可以使用直接行处理实现远程数据与任何中央数据源、应用程序或 Web 服务之间的通信。

本教程介绍如何将面向 Java 和 .NET 的 MobiLink 服务器 API 用于简单直接行处理。将客户端 RemoteOrders 表与统一数据库同步，并为 OrderComments 表添加直接行处理的特殊处理。

您可以为 RemoteOrders 表建立一个简单的同步。

### 必需的软件

- ◆ SQL Anywhere 10.0.0
- ◆ Java 软件开发工具包或 Microsoft .NET Framework

### 能力和经验

您需要：

- ◆ 熟悉 Java 或 .NET。
- ◆ 了解 MobiLink 事件脚本和 MobiLink 同步的基本知识。

### 目标

您将掌握并熟悉以下内容：

- ◆ 面向 Java 和 .NET 的 MobiLink 服务器 API。
- ◆ 创建适用于 MobiLink 直接行处理的方法。

### 建议阅读的背景知识

有关 MobiLink 同步的详细信息，请参见“[MobiLink 同步简介](#)”第 3 页。

有关同步技术的详细信息，请参见“[同步技术](#)”《[MobiLink - 服务器管理](#)》。

有关直接行处理的详细信息，请参见“[直接行处理](#)”《[MobiLink - 服务器管理](#)》。

## 第 1 课：建立 MobiLink 统一数据库

可以将 SQL Anywhere、Adaptive Server Enterprise、Oracle、Microsoft SQL Server 或 IBM DB2 MobiLink 统一数据库用作统一数据库。MobiLink 统一数据库是数据的中央存储库，包括用来管理同步过程的 MobiLink 系统表和存储过程。使用直接行处理，可与统一数据库以外的其它数据源同步，但您仍需要统一数据库来维护 MobiLink 服务器所使用的信息。

在本课中，您将：

- ◆ 创建数据库并定义 ODBC 数据源。
- ◆ 添加数据表以同步远程客户端。
- ◆ 安装 MobiLink 系统表和存储过程。

### 注意

如果您已建立了具有 MobiLink 系统对象和 DSN 的 MobiLink 统一数据库，则可跳过本课。

### 创建统一数据库

在本教程中，您将使用 Sybase Central 的 [创建数据库向导] 创建 SQL Anywhere 数据库。

#### ◆ 创建 SQL Anywhere RDBMS

1. 启动 Sybase Central。  
选择 [开始] ► [程序] ► [SQL Anywhere 10] ► [Sybase Central]。  
即会显示 Sybase Central。
2. 在 Sybase Central 中，选择 [工具] ► [SQL Anywhere 10] ► [创建数据库]。  
即会出现 [创建数据库向导]。
3. 单击 [下一步]。
4. 保留缺省值 [在这台计算机上创建数据库]，然后单击 [下一步]。
5. 输入数据库文件名和路径。例如，输入：  
`c:\MLdirect\MLconsolidated.db`
6. 请按照向导中的其余说明进行操作，除下面的项外，均接受缺省值。
  - ◆ 取消选中 [安装 jConnect 元信息支持] 选项。
  - ◆ 取消选中 [最后一次连接断开后停止数据库] 选项。
7. 单击 [完成]。

在 Sybase Central 中就会出现名为 MLconsolidated 的数据库。

### 为统一数据库定义 ODBC 数据源。

使用 SQL Anywhere 10 驱动程序为 MLconsolidated 数据库定义 ODBC 数据源。

#### ◆ 为统一数据库定义 ODBC 数据源

1. 启动 ODBC 管理器：

在 Sybase Central [工具] 菜单中，选择 [SQL Anywhere 10] ► [打开 ODBC 管理器]。  
即会出现 [ODBC 数据源管理器]。

2. 在 [用户 DSN] 选项卡上，单击 [添加]。

即会出现 [创建新数据源] 对话框。

3. 选择 [SQL Anywhere 10]，然后单击 [完成]。

此时会出现 [SQL Anywhere 10 的 ODBC 配置] 对话框。

4. 在 [ODBC] 选项卡上，键入数据源名 **mobilink\_db**。在 [登录] 选项卡上，在 [用户 ID] 中键入 **DBA**，在 [口令] 中键入 **sql**。在 [数据库] 选项卡上，在 [服务器名] 中键入 **MLconsolidated**，在 [数据库文件] 中键入 **c:\MLdirect\MLconsolidated.db**。

5. 单击 [确定]，定义数据源，再次单击 [确定] 关闭 [ODBC 管理器]。

### 为同步创建表

在此过程中，在 MobiLink 统一数据库中创建 RemoteOrders 表。RemoteOrders 表包含以下各列：

列	说明
order_id	订单的唯一标识符。
product_id	产品的唯一标识符。
quantity	销售项目的数量。
order_status	订单状态。
last_modified	行上次修改的日期。此列用于基于时间戳的下载，这是一种为提高同步效率而过滤行的常用技术。

#### ◆ 创建 RemoteOrders 表

1. 使用 Interactive SQL 连接到数据库。

从 Sybase Central 或命令提示符可以启动 Interactive SQL。

- ◆ 若要从 Sybase Central 启动 Interactive SQL，请单击数据库 **MLconsolidated - DBA**。从 Sybase Central [文件] 菜单中，选择 [打开 Interactive SQL]。
- ◆ 若要在命令提示符下启动 Interactive SQL，请键入以下命令：

```
dbisql -c "dsn=mobilink_db"
```

2. 在 Interactive SQL 中运行以下命令创建 RemoteOrders 表。

```
CREATE TABLE RemoteOrders
(
  order_id          integer not null,
  product_id       integer not null,
  quantity         integer,
  order_status     varchar(10) default 'new',
  last_modified    timestamp default current timestamp,
  primary key(order_id)
)
```

Interactive SQL 会在统一数据库中创建 RemoteOrders 表。

在 Interactive SQL 中为以下过程保持连接。

### 运行 MobiLink 安装脚本

可以在 SQL Anywhere 10 安装目录的 *MobiLink/setup* 子目录中查找每个支持的统一数据库（SQL Anywhere、Adaptive Server Enterprise、Oracle、Microsoft SQL Server 和 IBM DB2）的安装脚本。

在此过程中建立 SQL Anywhere 统一数据库。可以使用 *syncsa.sql* 安装脚本来执行这一操作。运行 *syncsa.sql* 时会创建一系列以 *ml\_* 为前缀的系统表和存储过程。MobiLink 服务器在同步过程中会使用这些表和存储过程。

#### ◆ 安装 MobiLink 系统表

1. 如果您尚未建立连接，则在 Interactive SQL 中连接到统一数据库。

在命令提示符处，键入以下命令：

```
dbisql -c "dsn=mobilink_db"
```

2. 在 Interactive SQL 中运行以下命令来创建 MobiLink 系统表和存储过程。用 SQL Anywhere 10 安装的位置来替换 *c:\Program Files\SQL Anywhere 10\*。

```
read "c:\Program Files\SQL Anywhere 10\MobiLink\setup\syncsa.sql"
```

Interactive SQL 将 *syncsa.sql* 应用到您的统一数据库。

在 Interactive SQL 中为下一课保持连接。

### 进一步阅读

有关创建 SQL Anywhere 数据库的信息，请参见“初始化实用程序 (dbinit)”一节《SQL Anywhere 服务器 - 数据库管理》。

有关创建表的信息，请参见“CREATE TABLE 语句”一节《SQL Anywhere 服务器 - SQL 参考》。

有关建立 MobiLink 统一数据库的信息，请参见“MobiLink 统一数据库”《MobiLink - 服务器管理》。

## 第 2 课：添加同步脚本

在本课中，您将为 SQL 行处理和直接行处理，将脚本添加到统一数据库中。

### SQL 行处理

SQL 行处理允许您将远程数据与 MobiLink 统一数据库中的表同步。基于 SQL 的脚本定义：

- ◆ 如何将来自 MobiLink 客户端上载的数据应用到统一数据库。
- ◆ 应从统一数据库下载哪些数据。

在本课中，您将为以下基于 SQL 的上载和下载事件编写同步脚本。

- ◆ **upload\_insert** 对插入远程客户端数据库的新订单如何应用到统一数据库进行定义。
- ◆ **download\_cursor** 对 MobiLink 统一数据库中更新的哪些订单应下载到远程客户端进行定义。

在此过程中，可以使用存储过程将同步脚本信息添加到 MobiLink 统一数据库。

#### ◆ 将基于 SQL 的脚本添加到 MobiLink 系统表

1. 如果您尚未建立连接，则在 Interactive SQL 中连接到统一数据库。

在命令提示符处，键入以下命令：

```
dbisql -c "dsn=mobilink_db"
```

2. 使用 ml\_add\_table\_script 存储过程为 upload\_insert 和 download\_cursor 事件添加基于 SQL 的表脚本。

在 Interactive SQL 中运行以下命令。upload\_insert 脚本将上载的 order\_id、product\_id、quantity 和 order\_status 插入到 MobiLink 统一数据库。download\_cursor 脚本使用基于时间戳的过滤将更新的行下载到远程客户端。

```
CALL ml_add_table_script( 'default', 'RemoteOrders',  
    'upload_insert',  
    'INSERT INTO RemoteOrders( order_id, product_id, quantity,  
order_status)  
    VALUES( ?, ?, ?, ? )' );
```

```
CALL ml_add_table_script( 'default', 'RemoteOrders',  
    'download_cursor',  
    'SELECT order_id, product_id, quantity, order_status  
    FROM RemoteOrders WHERE last_modified >= ?');
```

```
commit
```

### 直接行处理处理

在本教程中，您将使用直接行处理将特殊处理添加到基于 SQL 的同步系统中。在此过程中，您将注册与 handle\_UploadData、handle\_DownloadData 和 end\_download 事件对应的方法名。在“[第 3 课：为处理直接行处理编写 Java 或 .NET 逻辑](#)”一节第 135 页中创建您自己的 java 或 .NET 类。

#### ◆ 在 MobiLink 系统表中添加直接行处理的信息

1. 在 Interactive SQL 中，连接到统一数据库：

在命令提示符处，键入以下命令：

```
dbisql -c "dsn=mobilink_db"
```

2. 注册用于 end\_download 事件的 Java 或 .NET 方法。

当触发 end\_download 连接事件时，使用此方法释放 i/o 资源。

对于 Java，在 Interactive SQL 中执行以下命令。

```
CALL ml_add_java_connection_script( 'default',
  'end_download',
  'MobiLinkOrders.EndDownload' );
```

对于 .NET，在 Interactive SQL 中执行以下命令。

```
CALL ml_add_dnet_connection_script( 'default',
  'end_download',
  'MobiLinkOrders.EndDownload' );
```

Interactive SQL 注册用于 end\_download 事件的用户定义的 EndDownload 方法。

3. 注册用于 handle\_UploadData 和 handle\_DownloadData 同步事件的 Java 或 .NET 方法。

对于 Java，在 Interactive SQL 中执行以下命令。

```
CALL ml_add_java_connection_script( 'default',
  'handle_UploadData',
  'MobiLinkOrders.GetUpload' );
```

```
CALL ml_add_java_connection_script( 'default',
  'handle_DownloadData',
  'MobiLinkOrders.SetDownload' );
```

```
commit
```

对于 .NET，在 Interactive SQL 中执行以下命令。

```
CALL ml_add_dnet_connection_script( 'default',
  'handle_UploadData',
  'MobiLinkOrders.GetUpload' );
```

```
CALL ml_add_dnet_connection_script( 'default',
  'handle_DownloadData',
  'MobiLinkOrders.SetDownload' );
```

```
commit
```

Interactive SQL 分别注册用于 handle\_UploadData 和 handle\_DownloadData 事件的用户定义的 GetUpload 和 SetDownload 方法。

#### 进一步阅读

有关使用基于 SQL 的事件将数据从远程客户端上载到 MobiLink 统一数据库的信息，请参见：

- ◆ “编写用于上载行的脚本”一节 《MobiLink - 服务器管理》
- ◆ “upload\_insert 表事件”一节 《MobiLink - 服务器管理》
- ◆ “upload\_update 表事件”一节 《MobiLink - 服务器管理》
- ◆ “upload\_delete 表事件”一节 《MobiLink - 服务器管理》

有关将数据上载到除统一数据库之外的数据源的信息，请参见“处理直接上载”一节 《MobiLink - 服务器管理》。

有关使用基于 SQL 的事件从 MobiLink 统一数据库下载数据的信息，请参见：

- ◆ “编写用于下载行的脚本”一节 《MobiLink - 服务器管理》
- ◆ “download\_cursor 表事件”一节 《MobiLink - 服务器管理》
- ◆ “download\_delete\_cursor 表事件”一节 《MobiLink - 服务器管理》

有关将数据下载到除统一数据库之外的数据源的信息，请参见“设置直接下载”一节 《MobiLink - 服务器管理》。

有关同步事件序列的信息，请参见“MobiLink 事件概述”一节 《MobiLink - 服务器管理》。

有关下载过滤的同步技术的信息，请参见“基于时间戳的下载”一节 《MobiLink - 服务器管理》和“在远程数据库之间对行进行分区”一节 《MobiLink - 服务器管理》。

有关管理脚本的信息，请参见“添加和删除脚本”一节 《MobiLink - 服务器管理》。

有关直接行处理的信息，请参见“直接行处理” 《MobiLink - 服务器管理》。

## 第 3 课：为处理直接行处理编写 Java 或 .NET 逻辑

在本课中，您将在客户端数据库的 OrderComments 表中使用直接行处理来处理行。为直接行处理添加以下方法：

- ◆ **GetUpload** 将此方法用于 handle\_UploadData 事件。GetUpload 将上载的注释写入名为 *orderComments.txt* 的文件中。
- ◆ **SetDownload** 将此方法用于 handle\_DownloadData 事件。SetDownload 使用 *orderResponses.txt* 文件来下载对远程客户端的响应。

还可以添加 EndDownload 方法来处理 end\_download 事件。

以下过程介绍如何创建 Java 或 .NET 类，其中包括用于处理的方法。有关完整列表，请参见“完整的 MobiLinkOrders 列表 (Java)”一节第 141 页或“完整的 MobiLinkOrders 列表 (.NET)”一节第 143 页。

### ◆ 为直接行处理创建 Java 或 .NET 类

1. 使用 Java 或 .NET 创建名为 MobiLinkOrders 的类。

对于 Java，在文本编辑器或开发环境中键入以下代码。

```
import ianywhere.ml.script.*;
import java.io.*;
import java.sql.*;

public class MobiLinkOrders{

    // to do...
```

对于 .NET，使用以下代码：

```
using iAnywhere.MobiLink.Script;
using System.IO;
using System.Data;
using System.Text;

public class MobiLinkOrders

    // to do...
```

2. 声明一个类级别 DBConnectionContext 实例。

在 Java 中：

```
// class level DBConnectionContext
DBConnectionContext _cc;
```

在 .NET 中：

```
// class level DBConnectionContext
private DBConnectionContext _cc = null;
```

MobiLink 服务器会将 DBConnectionContext 实例传递到类构造函数。DBConnectionContext 封装有关与 MobiLink 统一数据库的当前连接的信息。

3. 创建类构造函数。

类构造函数设置类级别 DBConnectionContext 实例。

对于 Java:

```
public MobiLinkOrders( DBConnectionContext cc )
{
    // set your class-level DBConnectionContext
    _cc = cc;
}
```

对于 .NET:

```
public MobiLinkOrders( DBConnectionContext cc )
{
    _cc = cc;
}
```

4. 声明用于文件输入和输出的对象。

对于 Java, 声明 java.io.FileWriter 和 java.io.BufferedReader:

```
// java objects for file i/o
FileWriter my_writer;
BufferedReader my_reader;
```

对于 .NET, 声明流写入器和流读取器:

```
// instances for file I/O
private static StreamWriter my_writer = null;
private static StreamReader my_reader = null;
```

5. 编写 EndDownload 方法。

此方法处理 end\_download 连接事件并提供释放资源的机会。

对于 Java:

```
public void EndDownload() throws IOException
{
    // free i/o resources
    if (my_reader!=null) my_reader.close();
    if (my_writer!=null) my_writer.close();
}
```

对于 .NET:

```
public void EndDownload()
{
    if( my_writer != null ) {
        my_writer.Close();
        my_writer = null;
    }
}
```

6. 编写 GetUpload 方法

GetUpload 方法获得表示 OrderComments 表的 UploadedTableData 类实例。OrderComments 表包含由远程销售雇员进行的特殊注释。您将在“第 5 课: 建立 MobiLink 客户端”一节第 147 页

中创建该表。UploadedTableData getInserts 方法返回新订单注释的结果集。writeOrderComment 方法将结果集中的每行写出至文本文件。

对于 Java:

```
//method for the handle UploadData synchronization event
public void GetUpload( UploadData ut ) throws SQLException, IOException
{
    // get an UploadedTableData for OrderComments
    UploadedTableData orderCommentsTbl = ut.getUploadedTableByName
("OrderComments");

    // get inserts uploaded by the MobiLink client
    ResultSet insertResultSet = orderCommentsTbl.getInserts();

    while( insertResultSet.next() ) {

        // get order comments
        int _commentID = insertResultSet.getInt("comment_id");
        int _orderID = insertResultSet.getInt("order_id");
        String _specialComments = insertResultSet.getString
("order_comment");

        if ( _specialComments != null)
        {
            writeOrderComment(_commentID,_orderID,_specialComments);
        }
    }
    insertResultSet.close();
}

// writes out comment details to file
public void writeOrderComment( int _commentID, int _orderID, String
_comments )
throws IOException
{
    // a FileWriter for writing order comments
    if(my_writer == null)
    {
        my_writer = new FileWriter( "C:\\MLdirect\\
\\orderComments.txt",true);
    }

    // write out the order comments to remoteOrderComments.txt
    my_writer.write(_commentID + "\\t" + _orderID + "\\t" + _comments);
    my_writer.write("\\n" );
    my_writer.flush();
}
}
```

在 .NET 中:

```
// method for the handle UploadData synchronization event.
public void GetUpload( UploadData ut )
{
    // get UploadedTableData for remote table called OrderComments
    UploadedTableData order_comments_table_data =
ut.GetUploadedTableByName( "OrderComments");

    // get inserts upload by the MobiLink client
```

```

        IDataReader new_comment_reader = order_comments_table_data.GetInserts
        ();

        while( new_comment_reader.Read() ) {
            // columns are
            // 0 - "order_comment"
            // 1 - "comment_id"
            // 2 - "order_id"
            // you can look up these values using the DataTable returned by:
            // order_comments_table_data.GetSchemaTable() if the send column
names
            // option is turned on on the remote.
            // in this example you just use the known column order to determine
the column
            // indexes

            // only process this insert of the order comment is not null
            if( !new_comment_reader.IsDBNull( 2 ) ) {
                int comment_id = new_comment_reader.GetInt32( 0 );
                int order_id = new_comment_reader.GetInt32( 1 );
                string comments= new_comment_reader.GetString( 2 );
                WriteOrderComment( comment_id, order_id, comments );
            }
        }
        // always close the reader when you are done with it!
        new_comment_reader.Close();
    }
}

```

## 7. 编写 SetDownload 方法:

- a. 获得表示 OrderComments 表的类实例。

使用 DBConnectionContext getDownloadData 方法可获得 DownloadData 实例。使用 DownloadData getDownloadTableByName 方法可为 OrderComments 表返回 DownloadTableData 实例。

对于 Java:

```

DownloadData download_d = _cc.getDownloadData();
DownloadTableData download_td = download_d.getDownloadTableByName
( "OrderComments" );

```

对于 .NET:

```

DownloadTableData comments_for_download =
    _cc.GetDownloadData().GetDownloadTableByName( "OrderComments" );

```

**注意:**

在“第 5 课：建立 MobiLink 客户端”一节第 147 页中的远程数据库上创建此表。

- b. 获得允许您将插入或更新操作添加到下载的准备好的语句或者 IDbCommand。

对于 Java，使用 DownloadTableData getUpsertPreparedStatement 方法来返回一个 java.sql.PreparedStatement 实例。

```

PreparedStatement update_ps = download_td.getUpsertPreparedStatement
();

```

对于 .NET，使用 DownloadTableData GetUpsertCommand 方法:

```
// you will add upserts to the set of operation that are going to be
// applied at the
// remote database
IDbCommand comments_upsert = comments_for_download.GetUpsertCommand
();
```

- c. 将下载响应发送到远程客户端。

在 `c:\MLdirect` 中创建名为 `orderResponses.txt` 的文本文件。此文件包含对注释的响应。例如，`orderResponses.txt` 可以包含以下含有表示 `comment_id`、`order_id` 和 `order_comment` 的制表符分隔的值的条目。

```
...
786 34 OK, we will ship promotional material.
787 35 Yes, the product is going out of production.
788 36 No, we can't increase your commission...
...
```

- d. 设置每行的下载数据。

对于 Java，以下示例遍历 `orderResponses.txt` 并将数据添加到 MobiLink 下载。

对于 Java:

```
// a BufferedReader for writing out responses
if (my_reader==null)
    my_reader = new BufferedReader(new FileReader( "c:\\MLdirect\\
\\orderResponses.txt"));

// send updated comments down to clients
String commentLine;

// read the first line from orderResponses.txt
commentLine = my_reader.readLine();

    while(commentLine != null)
    {
        // get the next line from orderResponses.txt
        String[] response_details = commentLine.split("\t");

        if (response_details.length != 3)
            {
                System.err.println("Error reading from
orderResponses.txt");
                System.err.println("Error setting direct row handling
download");
                return;
            }

        int comment_id = Integer.parseInt(response_details[0]);
        int order_id = Integer.parseInt(response_details[0]);
        String updated_comment = response_details[2];

        // set an order comment response in the MobiLink download
        update_ps.setInt(1, comment_id);
        update_ps.setInt(2, order_id);
        update_ps.setString(3, updated_comment);
        update_ps.executeUpdate();

        // get next line from orderResponses.txt
        commentLine = my_reader.readLine();
    }
```

对于 .NET:

```
string comment_line;
while( (comment_line = my_reader.ReadLine()) != null) {
    // three values are on each line separated by '\t'
    string[] response_details = comment_line.Split( '\t' );
    if( response_details.Length != 3 ) {
        throw( new SynchronizationException( "Error reading from
orderResponses.txt" ) );
    }
    int comment_id = System.Int32.Parse( response_details[0] );
    int order_id = System.Int32.Parse( response_details[1] );
    string comments= response_details[2];

    // Parameters of the correct number and type have already been
added
    // so you just need to set the values of the IDataParameters
    ((IDataParameter)(comments_upsert.Parameters[0])).Value =
comment_id;
    ((IDataParameter)(comments_upsert.Parameters[1])).Value =
order_id;
    ((IDataParameter)(comments_upsert.Parameters[2])).Value =
comments;
    // add the upsert operation
    comments_upsert.ExecuteNonQuery();
}
```

- e. 关闭用来将插入操作或更新操作添加到下载的准备好的语句。

对于 Java:

```
update_ps.close();
```

对于 .NET, 不必关闭 `IDBCommand`。MobiLink 会在下载结束时将其消灭。

8. 编译您的类文件。

- ◆ 浏览到包含 Java 或 .NET 源文件的目录。
- ◆ 编译 `MobiLinkOrders` 时引用面向 Java 或 .NET 的 `MobiLink` 服务器 API 库。

对于 Java, 您需要引用 SQL Anywhere 安装目录的 `java` 子目录中的 `mlscript.jar`。键入以下命令行来编译 Java 类, 用您的 SQL Anywhere 10 目录替换 `c:\Program Files\SQL Anywhere 10`。

```
javac -classpath "c:\Program Files\SQL Anywhere 10\java\mlscript.jar"
MobiLinkOrders.java
```

对于 .NET, 使用以下命令:

```
csc /out:MobiLinkServerCode.dll /target:library /reference:"c:\Program
Files\SQL Anywhere 10\Assembly\v1\iAnywhere.MobiLink.Script.dll"
MobiLinkOrders.cs
```

## 进一步阅读

有关类构造函数和 `DBConnectionContext` 的详细信息, 请参见:

- ◆ .NET 同步逻辑: “构造函数”一节 [《MobiLink - 服务器管理》](#)

- ◆ Java 同步逻辑：“构造函数”一节 《MobiLink - 服务器管理》

有关 Java 同步逻辑的详细信息，请参见“使用 Java 语言编写同步脚本” 《MobiLink - 服务器管理》。

有关 .NET 同步逻辑的详细信息，请参见“使用 .NET 编写同步脚本” 《MobiLink - 服务器管理》。

有关直接行处理的详细信息，请参见“直接行处理” 《MobiLink - 服务器管理》。

## 完整的 MobiLinkOrders 列表 (Java)

以下是适用于 Java 直接行处理的完整的 MobiLinkOrders 列表。有关分步说明，请参见“第 3 课：为处理直接行处理编写 Java 或 .NET 逻辑”一节第 135 页。

```
import ianywhere.ml.script.*;
import java.io.*;
import java.sql.*;

public class MobiLinkOrders {

    // class level DBConnectionContext
    DBConnectionContext _cc;

    // java objects for file i/o
    FileWriter my_writer;
    BufferedReader my_reader;
    public MobiLinkOrders( DBConnectionContext cc )
        throws IOException, FileNotFoundException
    {
        // declare a class-level DBConnectionContext
        _cc = cc;
    }

    public void writeOrderComment( int _commentID, int _orderID, String
    _comments )
        throws IOException
    {
        if(my_writer == null)
            // a FileWriter for writing order comments
            my_writer = new FileWriter( "C:\\MLdirect\\
            \orderComments.txt",true);

        // write out the order comments to remoteOrderComments.txt
        my_writer.write(_commentID + "\t" + _orderID + "\t" + _comments);
        my_writer.write("\n" );
        my_writer.flush();

    }

    // method for the handle_UploadData synchronization event
    public void GetUpload( UploadData ut ) throws SQLException, IOException
    {
        // get an UploadedTableData for OrderComments
        UploadedTableData orderCommentsTbl = ut.getUploadedTableByName
        ("OrderComments");

        // get inserts uploaded by the MobiLink client
        ResultSet insertResultSet = orderCommentsTbl.getInserts();
```

```
while( insertResultSet.next() ) {
    // get order comments
    int _commentID = insertResultSet.getInt("comment_id");
    int _orderID = insertResultSet.getInt("order_id");
    String _specialComments = insertResultSet.getString("order_comment");

    if (_specialComments != null)
    {
        writeOrderComment(_commentID,_orderID,_specialComments);
    }
}
insertResultSet.close();
}

public void SetDownload() throws SQLException, IOException
{
    DownloadData download_d = _cc.getDownloadData();

    DownloadTableData download_td = download_d.getDownloadTableByName
( "OrderComments" );

    PreparedStatement update_ps = download_td.getUpsertPreparedStatement
();
    // a BufferedReader for writing out responses
    if (my_reader==null)
        my_reader = new BufferedReader(new FileReader( "c:\\MLdirect\\
\\orderResponses.txt"));

    // get the next line from orderResponses
    String commentLine;
    commentLine = my_reader.readLine();

    // send comment responses down to clients
    while(commentLine != null)
    {
// get the next line from orderResponses.txt
        String[] response_details = commentLine.split("\t");

        if (response_details.length != 3)
        {
            System.err.println("Error reading from orderResponses.txt");
            System.err.println("Error setting direct row handling
download");
            return;
        }
        int comment_id = Integer.parseInt(response_details[0]);
        int order_id = Integer.parseInt(response_details[1]);
        String updated_comment = response_details[2];

// set an order comment response in the MobiLink download
        update_ps.setInt(1, comment_id);
        update_ps.setInt(2, order_id);
        update_ps.setString(3, updated_comment);
        update_ps.executeUpdate();

        // get next line
        commentLine = my_reader.readLine();
    }

    update_ps.close();
}
```

```

    }
    public void EndDownload() throws IOException
    {
        // close i/o resources
        if (my_reader!=null) my_reader.close();
        if (my_writer!=null) my_writer.close();
    }
}

```

## 完整的 MobiLinkOrders 列表 (.NET)

以下是适用于 .NET 基于对象的上载和下载的完整的 MobiLinkOrders 列表。有关分步说明，请参见“第 3 课：为处理直接行处理编写 Java 或 .NET 逻辑”一节第 135 页。

```

using iAnywhere.MobiLink.Script;
using System.IO;
using System.Data;
using System.Text;

public class MobiLinkOrders
{
    // class level DBConnectionContext
    private DBConnectionContext _cc = null;

    // instances for file I/O
    private static StreamWriter my_writer = null;
    private static StreamReader my_reader = null;

    public MobiLinkOrders( DBConnectionContext cc )
    {
        _cc = cc;
    }
    public void WriteOrderComment( int comment_id,
        int order_id,
        string comments )
    {
        if( my_writer == null ) {
            my_writer = new StreamWriter( "c:\\mlobjbased\\orderComments.txt" );
        }
        my_writer.WriteLine( "{0}\t{1}\t{2}", comment_id, order_id, comments );
        my_writer.Flush();
    }
    // method for the handle UploadData synchronization event.
    public void GetUpload( UploadData ut )
    {
        // get UploadedTableData for remote table called OrderComments
        UploadedTableData order_comments_table_data = ut.GetUploadedTableByName
        ( "OrderComments" );

        // get inserts upload by the MobiLink client
        IDataReader new_comment_reader = order_comments_table_data.GetInserts();

        while( new_comment_reader.Read() ) {
            // columns are
            // 0 - "order_comment"
            // 1 - "comment_id"
            // 2 - "order_id"
            // you can look up these values using the DataTable returned by:
            // order_comments_table_data.GetSchemaTable() if the send column

```

```
names
    // option is turned on on the remote.
    // in this example you just use the known column order to determine
the column
    // indexes

    // only process this insert of the order comment is not null
    if( !new_comment_reader.IsDBNull( 2 ) ) {
        int comment_id = new_comment_reader.GetInt32( 0 );
        int order_id   = new_comment_reader.GetInt32( 1 );
        string comments= new_comment_reader.GetString( 2 );
        WriteOrderComment( comment_id, order_id, comments );
    }
}
// always close the reader when you are done with it!
new_comment_reader.Close();
}

private const string read_file_path = "c:\\mlobjbased\\
\\orderResponses.txt";

// method for the handle DownloadData synchronization event
public void SetDownload()
{
    if( (my_reader == null) && !File.Exists( read_file_path ) ) {
        System.Console.Out.Write( "Nothing to do in SetDownload() there is no
file to read." );
        return;
    }
    DownloadTableData comments_for_download =
        _cc.GetDownloadData().GetDownloadTableByName( "OrderComments" );

    // you will add upserts to the set of operation that are going to be
applied at the
    // remote database
    IDbCommand comments_upsert = comments_for_download.GetUpsertCommand();

    if( my_reader == null ) {
        my_reader = new StreamReader( read_file_path );
    }
    string comment_line;
    while( (comment_line = my_reader.ReadLine()) != null) {
        // three values are on each line separated by '\t'
        string[] response_details = comment_line.Split( '\t' );
        if( response_details.Length != 3 ) {
            throw( new SynchronizationException( "Error reading from
orderResponses.txt" ) );
        }
        int comment_id = System.Int32.Parse( response_details[0] );
        int order_id   = System.Int32.Parse( response_details[1] );
        string comments= response_details[2];

        // Parameters of the correct number and type have already been added
        // so you just need to set the values of the IDataParameters
        ((IDataParameter)(comments_upsert.Parameters[0])).Value = comment_id;
        ((IDataParameter)(comments_upsert.Parameters[1])).Value = order_id;
        ((IDataParameter)(comments_upsert.Parameters[2])).Value = comments;
        // add the upsert operation
        comments_upsert.ExecuteNonQuery();
    }
}

public void EndDownload()
{
}
```

```
    if( my_writer != null ) {  
        my_writer.Close();  
        my_writer = null;  
    }  
    if( my_reader != null ) {  
        my_reader.Close();  
        my_reader = null;  
    }  
}
```

## 第 4 课：启动 MobiLink 服务器

在本课中，您将启动 MobiLink 服务器。使用 `-c` 选项启动 MobiLink 服务器 (mlsrv10) 连接到统一数据库，并分别使用 `-sl java` 或 `-sl dnet` 选项装载 Java 或 .NET 类。

### ◆ 为直接行处理启动 MobiLink 服务器

- 连接到统一数据库，然后在 `mlsrv10` 命令行装载 .NET 或 Java 类。

对于 Java，键入以下命令。用 Java 源文件的位置替换 `c:\MLdirect`。

```
mlsrv10 -c "dsn=mobilink_db" -o serverOut.txt -v+ -dl -zu+ -x tcpip -sl java (-cp c:\MLdirect)
```

对于 .NET：

```
mlsrv10 -c "dsn=mobilink_db;uid=DBA;pwd=sql" -o serverOut.txt -v+ -dl -zu+ -x tcpip -sl dnet (-Dautoloadpath=c:\MLdirect)
```

即会出现 MobiLink 服务器窗口。

下面对本教程中使用的每个 MobiLink 服务器选项均加以说明。选项 `-o`、`-v` 和 `-dl` 提供调试和疑难解答信息。适宜在开发环境中使用这些记录选项。出于性能原因，`-v` 和 `-dl` 通常不在生产中使用。

选项	说明
<code>-c</code>	引出连接字符串。
<code>-o</code>	指定消息日志文件 <code>serverOut.txt</code> 。
<code>-v+</code>	<code>-v</code> 选项指定记录哪些信息。使用 <code>-v+</code> 设置最大详细记录。
<code>-dl</code>	在屏幕上显示所有日志消息。
<code>-zu+</code>	自动添加新用户。
<code>-x</code>	为 MobiLink 客户端设置通信协议和参数。
<code>-sl java</code>	指定一组目录以便用来搜索类文件，并强制在服务器启动时装载 Java 虚拟机。
<code>-sl dnet</code>	指定 .NET 程序集的位置，并强制在服务器启动时装载 CLR。

### 进一步阅读

有关 MobiLink 服务器选项的完整列表，请参见“[MobiLink 服务器选项](#)”《[MobiLink - 服务器管理](#)》。

有关装载 Java 和 .NET 类的详细信息，请分别参见“[-sl java 选项](#)”一节《[MobiLink - 服务器管理](#)》和“[-sl dnet 选项](#)”一节《[MobiLink - 服务器管理](#)》。

## 第 5 课：建立 MobiLink 客户端

在本教程中，您将针对统一数据库和 MobiLink 客户端使用 SQL Anywhere 数据库。用于教程目的时，MobiLink 客户端、统一数据库和 MobiLink 服务器均驻留在同一台计算机上。

若要建立 MobiLink 客户端数据库，请创建 RemoteOrders 和 OrderComments 表。RemoteOrders 表对应于统一数据库中的 RemoteOrders 表。MobiLink 服务器使用基于 SQL 的脚本与远程订单进行同步。OrderComments 表仅在客户端数据库上使用。MobiLink 服务器使用特殊事件处理 OrderComments 表。

也在客户端数据库上创建同步用户、同步发布和同步预订。

### ◆ 建立 MobiLink 客户端数据库

1. 创建 MobiLink 客户端数据库。

在本课中，您将使用 dbinit 命令行实用程序创建 SQL Anywhere 数据库。

- ◆ 要创建 SQL Anywhere 数据库，请在命令提示符处键入以下命令：

```
dbinit -i -k remotel
```

-i 和 -k 选项分别告诉 dbinit 忽略 Sybase jConnect 支持和 Watcom SQL 兼容性视图。

- ◆ 要启动数据库引擎，请键入：

```
dbeng10 remotel
```

2. 使用 Interactive SQL 连接到 MobiLink 客户端数据库。

从命令提示符处键入：

```
dbisql -c "eng=remotel;uid=DBA;pwd=sql"
```

3. 创建 RemoteOrders 表。

在 Interactive SQL 中执行以下命令：

```
create table RemoteOrders (
    order_id          integer not null,
    product_id       integer not null,
    quantity         integer,
    order_status      varchar(10) default 'new',
    primary key(order_id)
)
```

4. 在 Interactive SQL 中运行以下命令创建 OrderComments 表：

```
create table OrderComments (
    comment_id       integer not null,
    order_id         integer not null,
    order_comment    varchar (255),
    primary key(comment_id),
    foreign key (order_id) references
        RemoteOrders (order_id)
)
```

5. 创建 MobiLink 同步用户、同步发布和同步预订：

```
CREATE SYNCHRONIZATION USER ml_sales1;  
CREATE PUBLICATION order_publ (TABLE RemoteOrders, Table OrderComments);  
CREATE SYNCHRONIZATION SUBSCRIPTION TO order_publ FOR ml_sales1  
TYPE TCCIP ADDRESS 'host=localhost'
```

**注意：**

使用 CREATE SYNCHRONIZATION SUBSCRIPTION 语句中的 TYPE 子句和 ADDRESS 子句指定如何连接到 MobiLink 服务器。

可以使用发布来确定同步哪些数据。在本例中指定整个 RemoteOrders 和 OrderComments 表。

**进一步阅读**

有关创建 SQL Anywhere 数据库的信息，请参见“[初始化实用程序 \(dbinit\)](#)”一节 《SQL Anywhere 服务器 - 数据库管理》。

有关 MobiLink 客户端的信息，请参见“[MobiLink 客户端介绍](#)” 《MobiLink - 客户端管理》。

有关在客户端创建 MobiLink 对象的信息，请参见：

- ◆ “[CREATE SYNCHRONIZATION USER 语句 \[MobiLink\]](#)”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 SYNCHRONIZATION SUBSCRIPTION 语句 \[MobiLink\]](#)”一节 《SQL Anywhere 服务器 - SQL 参考》

## 第 6 课：同步

dbmsync 实用程序为 SQL Anywhere 远程数据库启动 MobiLink 同步。在启动 dbmsync 之前，将订单数据和注释添加到远程数据库。

### ◆ 建立远程数据（客户端）

1. 使用 Interactive SQL 连接到 MobiLink 客户端数据库：

在命令提示符处，键入

```
dbisql -c "eng=remotel;uid=DBA;pwd=sql"
```

2. 将订单添加到客户端数据库的 RemoteOrders 表中。

在 Interactive SQL 中执行以下命令。

```
INSERT INTO RemoteOrders (order_id, product_id, quantity, order_status)
VALUES (1,12312,10,'new')
```

3. 将注释添加到客户端数据库的 OrderComments 表中。

在 Interactive SQL 中执行以下命令。

```
INSERT INTO OrderComments (comment_id, order_id, order_comment)
VALUES (1,1,'send promotional material with the order')
```

4. 提交所做的更改。

在 Interactive SQL 中执行以下语句：

```
COMMIT;
```

### ◆ 启动同步客户端（客户端）

- 在命令提示符处，键入以下命令（全部在一行上）：

```
dbmsync -c "eng=remotel;uid=DBA;pwd=sql" -e scn=on -o rem1.txt -v+
```

以下是每个选项的说明：

选项	说明
-c	指定连接字符串。
-e scn	将 SendColumnNames 设置为 on。按名称引用列时，直接行处理需要这样设置。
-o	指定消息日志文件 <i>rem1.txt</i> 。
-v+	-v 选项指定记录哪些信息。使用 -v+ 设置最大详细记录。

一旦启动 MobiLink 同步客户端，即会显示一个输出屏幕，指示同步已成功。基于 SQL 的同步将客户端 RemoteOrders 表中的行传送到统一数据库中的 RemoteOrders 表。

Java 或 .NET 处理在 *orderComments.txt* 中插入注释。下一步，在 *orderResponses.txt* 中插入响应以下载到远程数据库。

◆ **使用直接行处理下载返回注释（服务器端和客户端）**

1. 插入返回注释。此操作在服务器端上进行。

将以下文本添加到 *orderResponses.txt* 中。必须使用制表符分隔条目。在行的结尾处，按 Enter 键。

```
1 1 Promotional material shipped
```

2. 使用 `dbmlsync` 客户端实用程序运行同步。

此操作在客户端上进行。

在命令提示符处，键入以下命令（全部在一行上）。

```
dbmlsync -c "eng=remotel;uid=DBA;pwd=sql" -o rem1.txt -v+ -e scn=on
```

出现 MobiLink 客户端实用程序。

在 Interactive SQL 中，从 `OrderComments` 表进行选择，以验证下载了行。

**注意**

使用直接行处理下载的行不会由 `mlsrv10 -v+` 选项输出，而会由远程 `-v+` 选项输出到远程日志中。

**进一步阅读**

有关 `dbmlsync` 的详细信息，请参见“[SQL Anywhere 客户端](#)” [《MobiLink - 客户端管理》](#)。

## 清除

从计算机中删除教程资料。

### ◆ 删除教程资料

1. 关闭 Interactive SQL 的所有实例。
2. 关闭 SQL Anywhere、MobiLink 和同步客户端窗口。
3. 删除所有与教程相关的 DSN:

- ◆ 启动 ODBC 管理器。

在命令提示符处键入以下命令:

```
odbcad32
```

- ◆ 删除 mobilink\_db 数据源。
4. 删除统一数据库和远程数据库:
    - ◆ 导航到统一数据库和远程数据库所在的目录。
    - ◆ 删除 *MLconsolidated.db*、*MLconsolidated.log*、*remote1.db* 和 *remote1.log*。

## 进一步阅读

有关 MobiLink 服务器 API 的详细信息，请参见：

- ◆ “使用 Java 语言编写同步脚本” 《MobiLink - 服务器管理》
- ◆ “使用 .NET 编写同步脚本” 《MobiLink - 服务器管理》

有关 MobiLink 直接行处理的详细信息，请参见“直接行处理” 《MobiLink - 服务器管理》。

---

# 索引

## 其它

### .NET

MobiLink 服务器 API 的优点, 18

MobiLink 教程, 103

### .NET 同步逻辑

关于, 18

## A

### authenticate\_user

Sybase Central 模型模式, 35

### AvantGo (见 M-Business Anywhere)

#### 安全性

MobiLink 概述, 20

## B

### 帮助

技术支持, xiv

### 表映射

关于 MobiLink, 28

在 MobiLink 模型模式中创建新的远程表, 28

### 并发

MobiLink 上载处理, 15

### 部署

MobiLink 模型批处理文件, 39

### 部署同步模型向导

关于, 38

## C

### CodeXchange

MobiLink 示例, 4

### Contact MobiLink 示例

Contact 表, 76

Customer 表, 74

Product 表, 77

SalesRep 表, 74

表, 71

构建, 69

关于, 68

监控统计信息, 80

用户, 73

运行, 69

### custase.sql

位置, 48

### custdb.db

SQL Anywhere CustDB 统一数据库, 48

### custdb.sqc

位置, 51

### CustDB MobiLink 示例

ULCustomer 表, 60

ULOrder 表, 58

ULProduct 表, 61

表, 54

用户, 57

### CustDB 应用程序

DB2, 48

MobiLink 示例应用程序, 45

同步脚本, 48

### customss.sql

位置, 48

### custora.sql

位置, 48

### 参照完整性

MobiLink 同步, 16

### 参照完整性和同步

MobiLink 客户端, 16

### 插件

MobiLink, 23

### 查找详细信息并提供反馈

技术支持, xiv

### 冲突解决

Contact 示例, 77

CustDB 示例, 61

### 创建同步模型

Sybase Central 任务, 24

### 创建同步模型向导

用法, 25

### 创建新远程表

MobiLink 模型模式, 29

### 错误

提供反馈, xiv

## D

### DB2

CustDB 教程, 48

### dbmlsync 实用程序

Mac OS X, 21

## F

### 反馈

提供, xiv

- 文档, xiv
- 分发的数据库
  - MobiLink 同步, 3
- 服务器启动的同步
  - 在模型模式中设置, 35
- 复制
  - (参见 MobiLink)

## G

- GUID
  - (参见 UUID)
- 概述
  - MobiLink, 4
- 更改您的统一数据库
  - 模型模式, 27
- 更新模式
  - MobiLink 向导, 39
- 故障
  - MobiLink 同步恢复, 15
- 挂接
  - (参见 事件挂接)
- 管理模式
  - Sybase Central 中的 MobiLink 插件, 28

## H

- 回退
  - MobiLink 警告, 14
- 获取帮助
  - 技术支持, xiv

## I

- iAnywhere 开发人员社区
  - 新闻组, xiv
- IBM DB2
  - CustDB 教程, 48
- IMAP 验证
  - MobiLink Sybase Central 模型模式, 35
- install-dir
  - 文档用法, xi

## J

- Java
  - MobiLink 服务器 API 的优点, 18
  - MobiLink 教程, 91
  - 同步逻辑, 18
- Java 同步逻辑

- 关于, 18
- 级联删除
  - MobiLink 同步, 16
- 技术支持
  - 新闻组, xiv
- 脚本
  - MobiLink 介绍, 13
  - MobiLink 模型模式, 35
- 教程
  - MobiLink .NET 逻辑, 103
  - MobiLink Contact 示例, 67
  - MobiLink CustDB 示例, 45
  - MobiLink Java 逻辑, 91
  - MobiLink 用于 Oracle, 81
  - MobiLink 直接行处理, 127
  - 使用 Java 或 .NET API 进行 MobiLink 自定义验证, 117

## K

- 开发人员社区
  - 新闻组, xiv
- 客户端
  - MobiLink 同步, 10
- 客户端事件挂接过程
  - (参见 事件挂接)
- 快速入门
  - MobiLink, 4

## L

- LDAP 验证
  - MobiLink Sybase Central 模型模式, 35
- 联机手册
  - PDF, viii

## M

- Mac OS X
  - MobiLink, 21
- MobiLink
  - .NET 教程, 103
  - Java 教程, 91
  - MobiLink CustDB 教程, 45
  - Oracle 教程, 81
  - 关于, 3
  - 过程概述, 12
  - 教程 - MobiLink 示例应用程序, 67
  - 客户端, 10
  - 快速入门, 4

---

- 模型模式, 28
  - 示例, 4
  - 特色, 4
  - 体系结构, 7
  - 同步基础知识, 3
  - 用于编写同步逻辑的选项, 18
- MobiLink 服务器
  - Mac OS X, 21
  - 入门, 4
- MobiLink 服务器 API
  - 优点, 18
- MobiLink 脚本
  - 关于, 13
- MobiLink 客户端
  - 关于, 10
- MobiLink 模型
  - 关于, 23
- MobiLink 模型简介
  - 关于, 24
- MobiLink 上载
  - 处理, 15
  - 已定义, 12
- MobiLink 事件
  - 介绍, 13
- MobiLink 特色
  - 关于, 4
- MobiLink 同步
  - .NET 教程, 103
  - custdb 示例数据库, 45
  - Java 教程, 91
  - 客户端, 10
- MobiLink 同步过程
  - 关于, 4
- MobiLink 同步简介
  - 关于, 3
- MobiLink 同步逻辑
  - .NET 教程, 103
  - Java 教程, 91
- MobiLink 下载
  - 已定义, 12
- MobiLink 直接行处理
  - 教程, 127
- 面向 .NET 的 MobiLink 服务器 API
  - 优点, 19
- 面向 Java 的 MobiLink 服务器 API
  - 优点, 18
- 模式

- MobiLink 模型重新部署, 39
- 模式更改
  - MobiLink 模型重新部署, 39
- 模型
  - MobiLink, 24
- 模型模式
  - 简介, 24
  - 用法, 28

## N

- newdb.bat
  - 位置, 48

## O

- Oracle
  - MobiLink 教程, 81

## P

- PDF
  - 文档, viii
- POP3 验证
  - MobiLink Sybase Central 模型模式, 35
- 批处理文件
  - MobiLink 模型部署, 39

## Q

- 全向同步 (见 双向同步)

## R

- 如何处理上载
  - 关于, 15
- 如何处理同步失败
  - MobiLink, 15
- 入门
  - MobiLink, 4
  - SyncConsole, 21

## S

- samples-dir
  - 文档用法, xi
- SQL Anywhere
  - 文档, viii
- SQL ROW\_DELETED\_TO\_MAINTAIN\_REFERENTIAL\_INTEGRITY
  - UltraLite 同步, 16
- SQL 同步逻辑

- 替代方法, 18
- Sybase Central
  - 管理模式, 28
  - 模型模式, 28
- SyncConsole
  - 入门, 21
- syncora.sql
  - 使用, 86
- 上载
  - MobiLink 处理, 15
  - MobiLink 定义, 12
  - MobiLink 事务, 14
- 设置
  - MobiLink 同步, 4
  - MobiLink 用创建同步模型向导, 25
  - 模型模式中服务器启动的同步, 35
- 使用 .NET 编写同步脚本
  - 教程, 103
- 使用 Java 编写同步脚本
  - 教程, 91
- 示例
  - Contact MobiLink 示例, 68
  - MobiLink, 4
  - MobiLink CustDB 应用程序, 45
- 示例数据库
  - MobiLink CustDB 应用程序, 45
- 示例应用程序
  - MobiLink CustDB 应用程序, 45
- 事件
  - MobiLink 介绍, 13
- 事件选项卡
  - MobiLink 模型模式, 34
- 事务
  - MobiLink 提交和回退, 14
  - 在 MobiLink 同步期间, 14
- 数据交换
  - MobiLink 同步, 3
- 数据库
  - 与 MobiLink 同步, 3
- 数据移动技术
  - MobiLink 同步, 3
- 死锁
  - MobiLink 上载处理, 15
- 碎片
  - (参见 分区)

## T

- 提交
  - MobiLink 警告, 14
- 通信故障
  - MobiLink 同步恢复, 15
- 同步
  - Java 教程, 91
  - MobiLink Oracle 教程, 81
  - MobiLink 过程概述, 12
  - MobiLink 事务, 14
  - MobiLink 系统的体系结构, 7
  - MobiLink 性能, 16
  - MobiLink 中的时间戳, 14
  - 关于 MobiLink, 3
  - 快速入门, 4
  - 用于编写同步逻辑的选项, 18
- 同步表
  - 在模型模式中添加映射, 28
- 同步过程
  - 关于, 12
- 同步过程中的事务, 14
- 同步基础知识
  - 关于, 3
- 同步技术
  - custdb 示例应用程序, 45
  - MobiLink Contact 示例教程, 67
- 同步脚本
  - .NET 教程, 103
  - Java 教程, 91
- 同步客户端
  - 用于 MobiLink 的 SQL Anywhere 或 UltraLite, 10
- 同步逻辑
  - 用于编写的选项, 18
- 同步模型
  - 简介, 24
- 同步上载
  - MobiLink 处理, 15
- 同步系统
  - 组件, 7
  - 同步系统的组成部分, 7
- 同步已部署模型
  - MobiLink 模型模式, 39
- 同步预订
  - (参见 预订)
- 统一数据库
  - MobiLink, 9

---

## 图标

手册中使用的, xii

## U

### upload\_delete

Contact 示例, 77

CustDB 示例, 61

## V

VB (见 Visual Basic)

## W

### 外部服务器

在 MobiLink 模型模式中验证, 35

完全同步 (见 双向同步)

### 文档

SQL Anywhere, viii

约定, ix

## X

### 下载

MobiLink 定义, 12

MobiLink 事务, 14

### 向导

MobiLink 部署同步模型, 39

MobiLink 创建同步模型, 25

MobiLink 更新模式, 39

在 MobiLink 模型模式中创建新的远程表和映射,  
28

### 向外部服务器验证

MobiLink Sybase Central 模型模式, 35

### 协议

MobiLink 同步, 7

### 新的表映射

MobiLink 模型模式中的对话框, 28

### 新闻组

技术支持, xiv

### 性能

MobiLink 上载处理, 16

### 修改表映射和同步选项

关于, 28

### 修改脚本

MobiLink 模型模式, 34

## Y

移动性 (见 MobiLink)

## 疑难解答

MobiLink 同步失败, 15

新闻组, xiv

### 映射

模型模式中的 MobiLink 表, 28

用于 Sybase Central 的 MobiLink 插件

关于, 23

用于编写同步逻辑的选项

关于, 18

### 远程表

在 MobiLink 模型模式中创建新远程表, 29

### 远程数据库

MobiLink 模型模式, 26

### 约定

文档, ix

文档中的文件名, xi

约束错误 (见 冲突)

## Z

在模型模式中设置服务器启动的同步

关于, 35

在模型模式中向外部服务器验证

MobiLink, 35

在模型模式中修改脚本

MobiLink, 34

在统一数据库上进行管理

Sybase Central 任务, 28

### 支持

新闻组, xiv

### 直接行处理

教程, 127

直接行访问 (见 直接行处理)

### 中央数据源

关于, 9

### 重新部署模型

MobiLink, 39

---