SQL Server 复制

基础指南

SQL Server 复制基础指南

微软 SQL Server 的复制功能是一项非常强大的技术，它可以实时地将数据从一个服务器移动到另一个，无论单向还是双向都可以。DBA 在进行 SQL Server 扩展时，也需要频繁地用到复制技术。在本次的技术手册中，我们将对 SQL Server 的复制技术进行一个详尽的介绍，包括复制的工作原理、相关配置以及不同类型的复制等。

SQL Server 复制技术原理

无论最终你是否希望成为一名 DBA，理解 SQL Server 的复制技术还是非常重要的。本部分将对 SQL Server 的复制原理进行一个简单介绍，通过阅读本文，相信你可以做出更加明智的决策。

- SQL Server 复制：工作原理
- SQL Server 复制：何时说不
- SQL Server 数据库复制教程（上）
- SQL Server 数据库复制教程（中）
- SQL Server 数据库复制教程（下）

合理利用 SQL Server 复制

在了解了 SQL Server 复制技术的基本原理之后，我们来看一下 SQL Server 复制在进行配置以及合理利用方面都有哪些注意事项，在本部分中，我们还讨论了利用 Service Broker 来代替 SQL Server 复制的方法。

- 如何设置 SQL Server 复制选项（上）
- 如何设置 SQL Server 复制选项（下）

- 利用 Service Broker 来代替复制
- SQL Server 升级后监控数据库镜像和复制

对比不同版本的复制选项

随着 SQL Server 不同版本的更新，SQL Server 复制选项也有着不同程度的变化，尽管从 SQL Server 2000 升级到 SQL Server 2005，数据复制功能的变化是根本性的，但从 SQL Server 2005 升级到 2008 的变化却是十分细微的。因此我们有必要了解一下不同版本复制功能的区别。

- SQL Server 2008 数据复制新特性及其带来的价值（上）
- SQL Server 2008 数据复制新特性及其带来的价值（下）
- SQL Server 2005 的复制存储过程选项
- 如何在升级到 SQL Server 2005 时复制数据库？

SQL Server 复制：工作原理

无论最终你是否希望成为一名 DBA，理解 SQL Server 复制还是非常重要的。本文将对 SQL Server 复制进行一个简单的介绍，通过阅读本文，相信你可以做出更加明智的决策。我曾经见过无数个复制项目进展得极为不顺利，因此为避免错误，最好的方法就是预先做出正确的决策。

什么是 SQL Server 复制?

简单地说，复制就是 SQL Server 将你的数据拷贝同时从多个地方获取过来的方式。复制的形式有所不同，以下是几种基本的复制形式：

快照复制。它就是一个简单的数据库拷贝。其实也并不简单，原始数据库需要被锁定，也就意味着快照生成过程中，这个数据库暂时不可用。这也确保了结果拷贝是独一无二的。

一个快照通常不会使用自身，也就是说它们可以作为其它形式复制的开始点。你为启动另外一个拷贝而进行了快照，然后可以使用其他形式的复制来对拷贝进行即时更新。

事务性复制。这是一种单向的复制，也就是说数据从一个主发布端流向一个或多个订阅端。在订阅端做出的任何修改都不会影响到发布端，因为数据流是单向的。行数据将不会被复制，取而代之，发布端事务日志中的事务将发送到每个订阅端，不断地重复这些事务，可以将数据拷贝进行即时更新。

更新订阅端的事务性复制。这是事务性复制的另一种形式，它允许订阅段将事务发送给发布端。数据之间也不会产生任何冲突，当订阅端数据库只更新数据子集的时候，这样的方式最佳。

合并复制。这是一种更加复杂的复制，它允许每一个参与的数据库拷贝都能成为发布端或订阅端，也就是说每一个数据库拷贝都是完全读写的。在应对数据冲突上，它需要特别的决议模块,不同地点的数据也是同时进行着更改。

(作者：Don Jones 译者：孙瑞 来源：TT 中国)

原文标题：SQL Server 复制：工作原理

链接：http://www.searchdatabase.com.cn/showcontent_29764.htm

SQL Server 复制：何时说不

试想一下，当你的事务性复制发布端有五个更新中的订阅端。订阅端 A 对一些数据进行了更改，并将事务发送给发布端。发布端运行事务并将更改应用到数据库拷贝中，结果这样的更改还要发送给另外四个订阅端。与此同时，订阅端 B、C、D、E 也进行同样的动作，这会导致什么样的结果?

在一个繁忙的数据库应用中，上面所描述的动作会迅速生成大量的流量， 即使网络连接全部用来进行这一任务，你同样会给 SQL Server 主机增加几何倍数的负载。每个服务器不仅需要跟上自己的变化，而且还要跟上其它服务器的变化。发布端还要负责合并所有的变更。

这么复杂的工作可行吗？当然，再某些情况下可以。但它并不是魔法，当超出网络与服务器负载能力时，它将停止工作。而修复复制更加痛苦，通常需要删除之前的复制，从第一步快照重新开始。

根据我的经验，复制对于分布在世界各地、处理着繁忙事务的数据库来说并不合适，它们都是由相对较慢的 WAN 来连接的。在大多数复制失败的情况下，我认为都是进行复制的人对其工作原理不甚了解。另外，如果你需要一个繁忙的数据库可以全球访问，云计算平台也是一个不错的选择，比如 Windows Azure。

(作者：Don Jones 译者：孙瑞 来源：TT 中国)

原文标题：SQL Server 复制：何时说不

链接：http://www.searchdatabase.com.cn/showcontent_29765.htm

SQL Server 数据库复制教程（上）

SQL Server 复制是一个包含在 Microsoft SQL Server 的软件包，它用于以实例间一种迁移一致状态进行服务器间的数据移动。

SQL Server 复制可以：

1、是单向的或者双向的；

2、是调度或实时传输的；

3、是将数据传输到一个实例或者多个实例。

复制拓扑

复制拓扑由三台服务器组成——订阅端、发行端和分发端。订阅端是接受数据的服务器。发行端是保持有可供订阅端使用的原始数据集的服务器。分发端是包含大量设置的服务器。同时，它还保持有从发行端传输给订阅端的数据。

目前有三种不同的复制技术可以使用。它们是快照复制、合并复制和事务复制。

快照复制是一个单向性数据传输。当更新数据从发行端向订阅端传输时，所有数据都是一次发送的。

合并复制是一个双向复制，它可以实时或通过调度来传输数据。合并复制是唯一可用的双向复制。

事务复制是从发行端向订阅端单向的。数据可以调度或实时发送。当数据向订阅端传输时，所有数据变化都以发行端设计的顺序进行处理。

当配置复制时，我们可以选择复制的对象。每个复制的数据对象都称为物件。物件可以是表格、视图、存储过程、函数、规则、数据类型等等。在选择数据库对象时，虽然并没有明确要求，但建议选择所有依赖的对象。同时，虽然也不要求选择子表，但必须确保可以复制表所使用的任意用户自定义数据类型以及规则。虽然数据类型可以手动传输到订阅端，但是它们必须存在于删除服务器，否则表格建立将无法实现。

当配置物件时，我们可以在物件配置过滤器。这些过滤器都是有效的 WHERE 子句，它们告诉 SQL Server 复制只传输表中的数据子集。我们可以任意使用表中的字段作为垂直过滤器的一部分。我们也可以使用水平过滤器。水平过滤器是一种打比方的说法，其实指的是：WHERE 子句应用于正在进行数据复制的物件，并且其中只有与 WHERE 子句相匹配的数据才会被传输到订阅端。

（作者：Denny Cherry 译者：曾少宁 来源：TT 中国）

原文标题：SQL Server 数据库复制教程（上）

链接：http://www.searchdatabase.com.cn/showcontent_24521.htm

SQL Server 数据库复制教程（中）

如何拓扑

在此，我们最主要的是要确定选择使用哪种复制拓扑。选择错误的拓扑将带来非常令人不满意的结果。

当需要一个临时的数据下发时，我们可以使用快照复制。因为每次快照下发时，所有的数据都是一次性移动的，但是它需要花费大量的带宽。只有在需要复制的数据变化远远超过了初始数据集的规模才在缓慢的 WAN 上使用快照复制。换言之，如果大部分的数据都在不断地更新，那么选择使用快照技术是非常正确的。如果并非如此，那么就不能选择它。

当需要在发行端和订阅端之间进行传输时，就应该选择合并复制。当有多个订阅端时，数据变化会及时地在网络上从发行端复制到所有的订阅端。

事务技术可能是复制最常见的形式。它是用来在几乎实时地（或按照调度）将数据变化传输到一个或多个订阅端。事务复制最常用于服务器之间的实时传输。

不管我们使用哪种 SQL Server 复制拓扑，订阅端之间都是完全独立的。订阅端可以因为多种原因而落后，包括网络拥挤、磁盘 I/O 拥挤以及用户进程锁定和阻塞。由于订阅端彼此都是独立的，因此，一个订阅端的速度下降并不影响其他的订阅端。

复制代理

数据是由复制代理传输的，其中每个发布都有几个复制代理需要创建。快照代理可以用在三种复制拓扑上。当复制创建后，就会得到物件快照并被上传到每个订阅端。当使用合并复制或事务复制时，将会有一个日志读取器代理在分发端上运行并捕捉事务日志的变化。然后这些变化会被记录到发行数据库上。

当使用事务复制时，数据变化将从发行数据库上读取并通过发行代理应用到订阅端上。当使用合并复制时，数据变化通过合并代理从发行数据库上读取。合并代理又将从订阅端上获取数据变化并上传到发行端以便分配到其他订阅端上。

当创建订阅时，我们有两个选项用于创建每个订阅端。我们可以使用发送订阅或请求订阅。当分配代理运行在发行数据库的服务器上工作时就是发送订阅。而当请求代理运行在订阅数据库的服务器上时就是请求订阅。

发送订阅的优点是分配代理集中在分发端上。这就能够限制管理开销同时保持订阅端的分配（或合并）代理低负载。请求订阅的优点是分配代理的工作量分散到所有订阅端上。当分配数据库位于同一台服务器或实例上作为发行端时，建议使用请求订阅。

（作者：Denny Cherry 译者：曾少宁 来源：TT 中国）

原文标题：SQL Server 数据库复制教程（中）

链接：http://www.searchdatabase.com.cn/showcontent_24522.htm

SQL Server 数据库复制教程（下）

复制获取

当建立分发端时，系统会提示选择一个文件夹来存储快照。当使用的都是发送订阅时，这可能是一个本地驱动器路径。当使用请求订阅或同时使用发送和请求订阅时，则必须指定一个网络共享路径。此网络共享不可以是一个管理共享，并且每个运行 SQL Server 代理的订阅端的域名帐号必须能读写网络共享。

如果运行的都是发送订阅而且有超过 30 个左右的订阅，那么在尝试启动订阅时将出现超时错误。最快速的补救方法就是编辑分发或合并代理的任务。编辑第二步并将任务类型修改为操作系统命令。接着将完整路径和可执行复制命令名称设置在现有参数的前面。SQL Server2008 的默认路径是 C:\Program Files\Microsoft SQL Server\100\COM（在 SQL 2000 中用 80 取代 100，而在 SQL 2005 则是 90 取代）。当运行分发代理时，可以使用 distrib.exe，而当运行合并代理时，则使用 replmerg.exe。

修复 SQL Server 复制故障可能会非常棘手。默认情况下，代理并不提供大量错误数据。我们可以通过修改任务属性中的-OutputVerboseLevel 开关来调整所接收到的错误数据总数。通过增加默认的数目，更多的错误数据将记录到任务步骤中。我们也可以终止运行代理的 SQL 代理任务，然后在 DOS 命令提示符中运行命令来轻松地看到更多错误数据。

当 SQL Server 复制有大量数据需要传输时，为了保持更新，需要一定数量的网络带宽。如果带宽无法满足，那么复制将越来越缓慢。如果在一个低带宽、低延迟的网络中，那么通过添加 SubscriptionStreams 开关（或者在已经存在开关的情况下，增加开关数目）将有助于增加线路数目。如果在一个高延迟网络，那么由于事务整合性是在流之间维护的，因此增加这些设置可能不会提高性能。

（作者：Denny Cherry 译者：曾少宁 来源：TT 中国）

原文标题：SQL Server 数据库复制教程（下）

链接：http://www.searchdatabase.com.cn/showcontent_24523.htm

如何设置 SQL Server 复制选项（上）

微软 SQL Server 的复制功能是一项非常强大的技术，它可以实时地将数据从一个服务器移动到另一个，无论单向还是双向都可以。

为确保 SQL Server 复制功能运行的高效性，你在配置工具时需要做出几个重要决策。

推还是拉？

这个策略决定了在哪里运行 Distribution Agent（或者 Merge Agent）。

默认情况下，SQL Server 的工作是由 SQL Agent 服务配置并运行的。这称作 push 设置，因为 SQL Agent 会将设置变化从 Distributor 那里推送给 Subscriber。如果 Agent 在 Subscriber 上面运行，它就称作 pull 方式订阅，因为数据将从 Distributor 那里拉过来。

当选择 Distribution Agent 或 Merge Agent 的存放位置时，你需要考虑以下几个方面。

如果你有多个 Subscriber 和唯一一个 Distributor，那你就应该使用 push 订阅。这将会减轻 CPU 和内存运行 Distribution Agent 或 Merge Agent 时的负载压力，把节省出来资源用在处理配置变更和用户请求上。

另外，如果你的 Publisher 和 Distributor 存放在物理服务器的同一位置，那么你就会在 Subscriber 上运行 Distribution Agent 或 Merge Agent。这将减轻 Publisher 的负载压力，实际上，Publisher 是 SQL Server 复制技术的关键服务器。

虽然所有服务器运行在同一个高速局域网情况下，以上的建议还是有效的，但是在广域网情况下进行复制数据，你可能还需要做一些改变。特别是在你有一个比较大的快照需要发送给 Subscriber 时。

SQL Server 复制功能允许你控制每次运行的线程数目，这样你就可以一次申请多个事务到你的 Subscriber 中。然而在使用 SQL Server 2005 及以下版本时，你会遭遇一个内置的性能瓶颈并会严重阻碍你的工作。

在 2005 和以前的版本中，Distribution Agent 同分布式数据库只有一个连接。因此，无论你在推送数据到 Subscriber 时运行多少个线程，数据传输也只能跟运行一个线程的速度一样。而 SQL Server 2008 则避免了此种情况的发生。

因此，如果在广域网，特别是低速广域网或者有高延迟的广域网中运行 SQL Server 复制功能时，Distribution Agent 从分布式数据库下载数据的速度将极为有限。

注意：当使用网络时，速度和延迟完全是两回事。

相反地，Distribution Agent 应该运行在 Distributor 上。这将允许单线程从 Distributor 那里将数据拉过来，同时多线程在 Subscriber 上交付那些事务。

（作者：Denny Cherry 译者：孙瑞 来源：TT 中国）

原文标题：如何设置 SQL Server 复制选项（上）

链接：http://www.searchdatabase.com.cn/showcontent_22850.htm

如何设置 SQL Server 复制选项（下）

快照

当在广域网中使用快照时，你可以遵循几个技巧来极大程度地缩减处理时间。

一般来说，你可能想要推翻技术。如果你要在广域网中使用快照功能，你会遭遇速度骤降。而且，在高延迟广域网和高负载环境中，你处理快照的时间可能比数据更改还要慢。

解决这一问题的最佳方法是设置推送订阅。当运行快照时，这个方法可以提供最快的数据处理速度。

当配置发布选项时，选择一个本地路径存储快照。然后启动 Distribution Agent 来创建快照。

当快照被创建，Distribution Agent 开始处理它后，停止 Distribution Agent。当 SQL Server Agent 任务停止时，浏览快照存储的文件夹。默认情况下是 C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\repldata\unc，配置不同存放的位置可能也会有所不同。

选择一种你习惯的压缩技术对文件和文件夹进行压缩。将压缩文件拷贝到 Subscriber 并在和 Distributor 一样的路径下解压缩数据。然后修改 SQL Server Agent 任务并拷贝所有参数带 SQL Server Agent 任务中。

在 Subscriber 上，打开命令提示符找到 COM 文件夹，输入 distrib.exe。C:\Program Files\Microsoft SQL Server\100\COM 是 SQL Server 2008 的默认存放位置。相应地，在 2005 版本和 2000 版本中，你把路径中的 100 改为 80 或 90 就可以。

将从 SQL Server Agent 中拷贝的 switch 粘贴到命令框内并按回车。Distribution Agent 将临时运行在 Subscriber 上，它将处理拷贝到 Subscriber 上的快照。

处理完快照之后，agent 将正常处理 Distributor 中的数据，你可以关闭窗口并在 Distributor 上启动 Distribution Agent。

注意你不可以在两台服务器上同时运行 Distribution Agent。在一个启动时，运行另一个就会报错并关闭。

SQL Server 复制功能可以成为一个非常强大的工具，但是在没有认真计划并充分准备的前提下，你很可能就无法完成相应的工作。

（作者：Denny Cherry 译者：孙瑞 来源：TT 中国）

原文标题：如何设置 SQL Server 复制选项（下）

链接：http://www.searchdatabase.com.cn/showcontent_22852.htm

利用 Service Broker 来代替复制

为什么使用服务代理

在开始探讨代码之前，我要讲一下为什么我选择使用 SQL 服务代理替代 SQL 复制的一点幕后原因。这样可能对阐述有好处。

(选择使用 SQL 服务代理代替 SQL 复制的)主要原因是我们想在数据从生产环境 OLTP 数据库迁移到报表数据库时，能对数据做 ETL(数据提取转换加载)。我们还需要很容易地就能从一个报表数据库扩展为多个数据库的能力，以满足报表服务器在不同地点的情况。

表

下面你会发现一些 SQL 代码，我们将用这些代码创建一些示例表。在我们的 OLTP 数据库中，我们创建了两个表，它们是表“LoanApplication”和表“Customer”，而在我们的报表数据库中我们只有一个表“LoanApplication”。当数据插入或者更新到表“LoanApplication ”和表“Customer”时，数据会打包成 XML 文档并发送到报表数据库。在我这里的示例代码中，所有东西都在一台服务器上，但是数据库要挪到分离的服务器上也是非常容易的。

SQL:

```
1. IF EXISTS (SELECT * FROM sys.DATABASES WHERE name =
   'Sample_OLTP')
2. DROP DATABASE Sample_OLTP
3. GO
4. IF EXISTS (SELECT * FROM sys.DATABASES WHERE name =
   'Sample_Reporting')
5. DROP DATABASE Sample_Reporting
6. GO
7.
8. CREATE DATABASE Sample_OLTP
9. CREATE DATABASE Sample_Reporting
10.    GO
11.
12.    ALTER DATABASE Sample_OLTP SET NEW_BROKER
13.    ALTER DATABASE Sample_Reporting SET NEW_BROKER
```

```
14.    GO
15.    ALTER DATABASE Sample_OLTP SET TRUSTWORTHY ON
16.    ALTER DATABASE Sample_Reporting SET TRUSTWORTHY ON
17.    GO
18.
19.    USE Sample_OLTP
20.    GO
21.    CREATE TABLE LoanApplication
22.    (ApplicationId INT IDENTITY(1,1),
23.    CreateTimestamp DATETIME,
24.    LoanAmount MONEY,
25.    SubmittedOn DATETIME,
26.    ApprovedOn DATETIME,
27.    LoanStatusId INT,
28.    PrimaryCustomerId INT,
29.    CoSignerCustomerId INT)
30.    GO
31.    CREATE TABLE Customer
32.    (CustomerId INT IDENTITY(1,1).
33.    FirstName VARCHAR(50),
34.    LastName VARCHAR(50),
35.    EmailAddress VARCHAR(255))
36.    GO
37.    USE Sample_Reporting
38.    GO
39.    CREATE TABLE LoanReporting
40.    (ApplicationId INT,
41.    CreateTimestamp DATETIME,
42.    LoanAmount MONEY,
43.    SubmittedOn DATETIME,
44.    ApprovedOn DATETIME,
45.    LoanStatusId INT,
46.    PrimaryCustomerId INT,
47.    PrimaryFirstName VARCHAR(50),
48.    PrimaryLastName VARCHAR(50),
49.    PrimaryEmailAddress VARCHAR(255),
50.    CoSignerCustomerId INT,
51.    CoSignerFirstName VARCHAR(50),
52.    CoSignerLastName VARCHAR(50),
53.    CoSignerEmailAddress VARCHAR(255))
```

```
54.     GO
```

服务代理对象

在这个系统中，我只利用一对服务代理队列来处理所有数据传输。这样可以在数据流入时维持相互一致性。这些 SQL 服务代理对象应该被在示例 OLTP 数据库和示例报表数据库中创建。

SQL:

```
1. CREATE MESSAGE TYPE ReplData_MT
2. GO
3. CREATE CONTRACT ReplData_Ct
4. (ReplData_MT SENT BY ANY)
5. GO
6. CREATE QUEUE ReplData_Source_Queue
7. GO
8. CREATE QUEUE ReplData_Destination_Queue
9. GO
10.     CREATE SERVICE ReplData_Source_Service
11.     ON QUEUE ReplData_Source_Queue
12.     (ReplData_Ct)
13.     GO
14.     CREATE SERVICE ReplData_Destination_Service
15.     ON QUEUE ReplData_Destination_Queue
16.     (ReplData_Ct)
17.     GO
```

路由(route)

在该 OLTP 数据库中，你创建了这样一个 route(根据你的实际服务器修改 BROKER_INSTANCE)。

SQL:

```
1. CREATE ROUTE ReplData_Route
2. WITH SERVICE_NAME='ReplData_Destination_Service',
3. BROKER_INSTANCE='566C7F7A-9373-460A-8BCC-5C1FD4BF49C9',
4. ADDRESS='LOCAL'
```

在该报表数据库中，你创建了这样一个 route(根据你的实际服务器修改 BROKER_INSTANCE)。

SQL:

```
1. CREATE ROUTE ReplData_Route
2. WITH SERVICE_NAME='ReplData_Source_Service',
3. BROKER_INSTANCE='A4EC5E44-60AF-4CD3-AAAD-C3D467AC682E',
4. ADDRESS='LOCAL'
```

OLTP 数据库中的存储过程

在该 OLTP 数据库中我们只需要一个存储过程。这个存储过程将处理信息的发送，这样我们就不必在每个表中写相同的代码。

SQL:

```
1. CREATE PROCEDURE SendTriggerData
2. @XMLData XML
3. AS
4. BEGIN
5. DECLARE @handle UNIQUEIDENTIFIER
6.
7. BEGIN DIALOG CONVERSATION @handle
8. FROM SERVICE ReplData_Source_Service
9. TO SERVICE 'ReplData_Destination_Queue'
10.     ON CONTRACT ReplData_Ct
11.     WITH ENCRYPTION=OFF;
12.
13.     SEND ON CONVERSATION @handle
14.     MESSAGE TYPE ReplData_MT
15.     (@XMLData)
16.     END
17.     GO
```

OLTP 数据库触发器

该 OLTP 数据库中每个表上的触发器被保持尽可能小，以便我们给该 OLTP 服务器增加尽可能少的负载负担。很明显我们会给该 OLTP 数据库带来额外负载，但是我们希望把它保持在最小。

SQL:

```
1. CREATE TRIGGER t_LoanApplication ON LoanApplication
2. FOR INSERT, UPDATE
3. AS
4. BEGIN
5. DECLARE @xml XML
6.
7. SET @xml = (SELECT *
8. FROM inserted AS LoanApplication
9. FOR XML AUTO, ROOT('root'))
10.
11.     EXEC SendTriggerData @xml
12.     END
13.     GO
14.
15.     CREATE TRIGGER t_Customer ON Customer
16.     FOR INSERT, UPDATE
17.     AS
18.     BEGIN
19.     DECLARE @xml XML
20.
21.     SET @xml = (SELECT *
22.     FROM inserted AS Customer
23.     FOR XML AUTO, ROOT('root'))
24.
25.     EXEC SendTriggerData @xml
26.     END
27.     GO
```

报表数据库中的存储过程

该报表数据库是实际工作执行的地方。在这里我们获取 XML 文档，识别数据从哪个表来，然后把 XML 文档传递给子存储过程，然后处理数据并更新表。

SQL:

```
1. CREATE PROCEDURE ProcessOLTPData_LoanApplication
2. @xml XML
3. AS
```

```
4. DECLARE  @hDoc INT
5. EXEC sp_xml_preparedocument @hDoc OUTPUT, @xml
6.
7. UPDATE LoanReporting
8. SET ApplicationId=a.ApplicationId,
9. CreateTimestamp = a.CreateTimestamp,
10.     LoanAmount=a.LoanAmount,
11.     SubmittedOn=a.SubmittedOn,
12.     ApprovedOn=a.ApprovedOn,
13.     LoanStatusId=a.LoanStatusId
14.     FROM OPENXML (@hDoc, '/root/LoanApplication')
15.     WITH (ApplicationId INT '@ApplicationId',
16.     CreateTimestamp DATETIME '@CreateTimestamp',
17.     LoanAmount MONEY '@LoanAmount',
18.     SubmittedOn DATETIME '@SubmittedOn',
19.     ApprovedOn DATETIME '@ApprovedOn',
20.     LoanStatusId INT '@LoanStatusId',
21.     PrimaryCustomerId INT '@PrimaryCustomerId',
22.     CoSignerCustomerId INT '@CoSignerCustomerId') a
23.     WHERE a.ApplicationId = LoanReporting.ApplicationId
24.
25.     INSERT INTO LoanReporting
26.     (ApplicationId, CreateTimestamp, LoanAmount, SubmittedOn,
   ApprovedOn, LoanStatusId, PrimaryCustomerId, CoSignerCustomerId)
27.     SELECT ApplicationId, CreateTimestamp, LoanAmount,
   SubmittedOn, ApprovedOn, LoanStatusId, PrimaryCustomerId,
   CoSignerCustomerId
28.     FROM OPENXML (@hDoc, '/root/LoanApplication')
29.     WITH (ApplicationId INT '@ApplicationId',
30.     CreateTimestamp DATETIME '@CreateTimestamp',
31.     LoanAmount MONEY '@LoanAmount',
32.     SubmittedOn DATETIME '@SubmittedOn',
33.     ApprovedOn DATETIME '@ApprovedOn',
34.     LoanStatusId INT '@LoanStatusId',
35.     PrimaryCustomerId INT '@PrimaryCustomerId',
36.     CoSignerCustomerId INT '@CoSignerCustomerId') a
37.     WHERE NOT EXISTS (SELECT * FROM LoanReporting WHERE
   a.ApplicationId = LoanReporting.ApplicationId)
38.
39.     EXEC sp_xml_removedocument @hDoc
```

```
40.    GO
41.    CREATE PROCEDURE PRocessOLTPData_Customer
42.    @xml XML
43.    AS
44.    DECLARE  @hDoc INT
45.    EXEC sp_xml_preparedocument @hDoc OUTPUT, @xml
46.
47.    UPDATE LoanReporting
48.    SET PrimaryEmailAddress = EmailAddress,
49.    PrimaryFirstName = FirstName,
50.    PrimaryLastName = LastName
51.    FROM OPENXML(@hDoc, '/root/Customer')
52.    WITH (CustomerId INT '@CustomerId',
53.    FirstName VARCHAR(50) '@FirstName',
54.    LastName VARCHAR(50) '@LastName',
55.    EmailAddress VARCHAR(255) '@EmailAddress') a
56.    WHERE PrimaryCustomerId = a.CustomerId
57.
58.    UPDATE LoanReporting
59.    SET CoSignerEmailAddress = EmailAddress,
60.    CoSignerFirstName = FirstName,
61.    CoSignerLastName = LastName
62.    FROM OPENXML(@hDoc, '/root/Customer')
63.    WITH (CustomerId INT '@CustomerId',
64.    FirstName VARCHAR(50) '@FirstName',
65.    LastName VARCHAR(50) '@LastName',
66.    EmailAddress VARCHAR(255) '@EmailAddress') a
67.    WHERE CoSignerCustomerId = a.CustomerId
68.
69.    EXEC sp_xml_removedocument @hDoc
70.    GO
71.
72.    CREATE PROCEDURE ProcessOLTPData
73.    AS
74.    DECLARE @xml XML
75.    DECLARE @handle UNIQUEIDENTIFIER, @hDoc INT
76.
77.    WHILE 1=1
78.    BEGIN
79.    SELECT @xml = NULL, @handle = NULL
```

```
80.
81.     WAITFOR (RECEIVE TOP (1) @handle=conversation_handle,
82.     @xml=CAST(message_body AS XML)
83.     FROM ReplData_Destination_Queue), TIMEOUT 1000
84.
85.     IF @handle IS NULL
86.     BREAK
87.
88.     EXEC sp_xml_preparedocument @hDoc OUTPUT, @xml
89.
90.     IF EXISTS (SELECT *
91.     FROM OPENXML(@hDoc, '/root/LoanApplication')
92.     )
93.     BEGIN
94.     EXEC ProcessOLTPData_LoanApplication @xml
95.     END
96.
97.     IF EXISTS (SELECT *
98.     FROM OPENXML (@hDoc, '/root/Customer')
99.     )
100.    BEGIN
101.    EXEC PRocessOLTPData_Customer @xml
102.    END
103.
104.    EXEC sp_xml_removedocument @hDoc
105.
106.    END CONVERSATION @handle
107.    END
108.    GO
```

(作者：Denny Cherry 译者：冯昀晖 来源：TT 中国)

原文标题：利用 Service Broker 来代替复制

链接：http://www.searchdatabase.com.cn/showcontent_34921.htm

SQL Server 升级后监控数据库镜像和复制

这篇文章是介绍数据库升级的案例记录一系列文章的最后一个技巧，它描述了如何将在 Windows 2000 服务器上运行的 SQL Server 2000 Active/Active 群集升级到 Windows Server2003 /SQL Server 2005 Active/Active 群集。咨询师 Matthew Schroeder 将通过 IT 世界和数据库管理团队的技术方面和决策过程方面进行详细阐述。这篇文章是基于两个在线升级：一个是商业网、另一个是 eBay 排序系统。由于考虑到机密原因，我们改变了实际方案的某些细节。

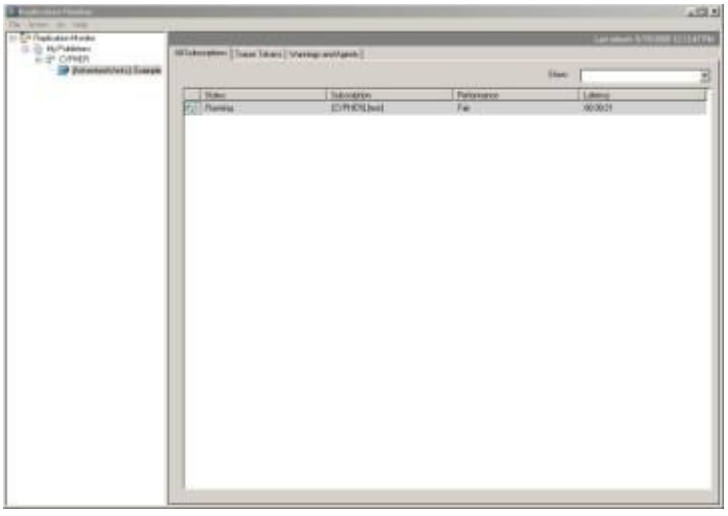
数据库管理员一般都从架构开始，这样就造成了他们对升级过程中如何监测 SQL Server 不清楚。到目前为止，升级最终要的部分就是进行计划，但是很少有公司资源筹划一次完整的负载升级测试。遗憾的是，当系统在产品的重压之下，SQL Server 升级中有效负载测试就开始进行了。

挑战

我们假设公司要么使用数据库镜像、复制，要么使用的是两个系统合并升级成的一个系统。我们检查一下如何监控复制以及数据库镜像保证所有的数据以有效可靠的方式进行传输：

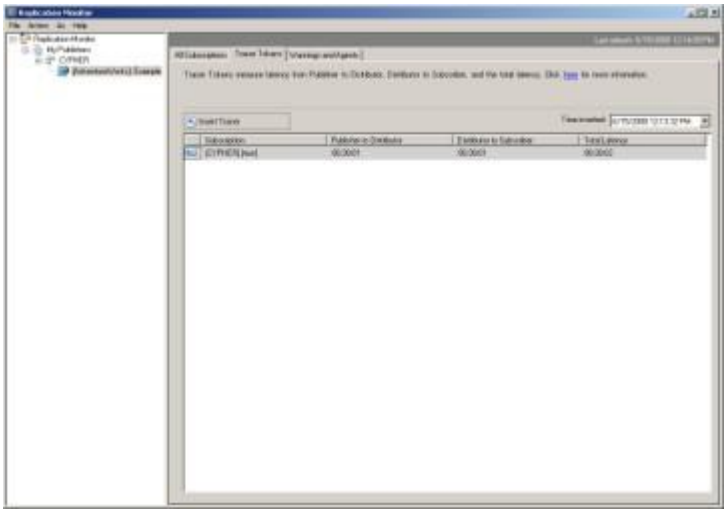
监控 SQL Server 复制

监控复制部分是为了确保所有在转换数据库中的的记录都能够被迁到目的数据库中，这要求你执行的操作就是点击“复制”和“启动复制监视器”，出现监视屏，如图一所示：



图一：点击“复制”和“启动复制监视器”。（点击放大图片）

这一屏幕的出现让你对执行结果有了初步的印象。那么“订阅”操作执行了吗？性能和滞后时间又如何呢？微软在 SQL Server 2005 上增加了一项非常好的性能：跟踪令牌（tracer token）。跟踪令牌主要是插入复制链里的数据包，（主要是让你能够为分发服务器测量时间（可以在同一台机器上或是它自身的 box），它还测量从分发服务器到订阅服务器的时间。（如图 2）



图二：SQL Server 2005 里的跟踪令牌为分发服务器测量时间而且还测量从分发服务器到订阅服务器的时间。（点击放大图片）

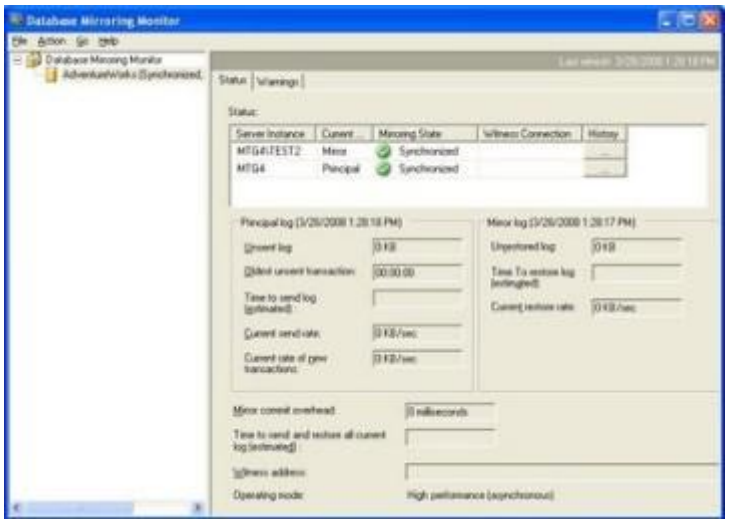
初始数据库快照进行订阅复制时，就会显示出一个感叹号。这并不是说出现了问题，而是在复制数据库快照时造成的加载。一旦这两个数据库同步了，感叹号就会消失，除非就是服务器真的出现超载现象。

如果感叹号还存在，你可能会使用跟踪令牌来检查是否发布服务器、分发服务器和订阅服务器出现了滞后时间。在这种状况下影响滞后时间的主要因素是 CPU、I/O、网络和 RAM。

如果这里的某一个方面实际上出现了性能减缓的现象，那就要切换 All Subscriptions 表、双击“订阅”获取错误表。这时候的祸首通常是架构、作业或是起始误差。

在 SQL Server 中监控数据库镜像

Monitoring Database Mirroring 不是很清楚，因为它没有单独的文件夹。如果设立了数据库镜像，就只需要点击主数据库的或者是镜像数据库、任务，然后选择 Launch Database Mirroring Monitor，就会出现图三中的屏幕。



图三：如果设置了数据库镜像，就选择 Launch Database Mirroring Monitor。（点击放大图片）

如果你已对有最近事务日志的镜像示图初始化，那这两个数据库应该在你到达如上图的屏幕之前就同步了。如果数据库镜像是为了确保高安全性而设置的，例如在镜像上面执行事务而不是主要数据库上执行，这一屏会让你了解管理费用是由“Mirror commit overhead”产生的。如果你才用了高可用性模式，它还会对于你镜像环境可能有多落后这一方面有很大的启发。

由于处于复制状态，性能问题在这些领域存在着同样的问题。所以如果在上面的这个屏幕上显示了任何问题，它们应该很容易被发现。

总结

这一系列的文章分为五个部分，我们通过以一些范例架构设计出对 OS/SQL Server 以及 SQL Server 动态应用程序升级计划。这些架构能够一起或单独使用。我们还介绍了如何监控数据库镜像以及监控数据库复制，这些都取决于你采用什么样的升级方式。

你应该通过升级定期回顾一下复制和数据库镜像性能。你可以在数据库镜像中设置警告，这些警告还可以被 SQL Server Agent 用来发送 DBA 警告或者 MOM/SCOM（MS 系统监控工具）以便于选取事件，还可以被它们用来给合适的群体发送警告。和 SQL Server Agent or MOM/SCOM 相比较，如果它本身就不发送警告，那么在复制时就很难有警告提示了。

（作者：Matthew Schroeder 译者：April 来源：TT 中国）

原文标题：SQL Server 升级后监控数据库镜像和复制

链接：http://www.searchdatabase.com.cn/showcontent_13842.htm

SQL Server 2008 数据复制新特性及其带来的价值（上）

尽管从 SQL Server 2000 升级到 SQL Server 2005，数据复制功能的变化是根本性的，但从 SQL Server 2005 升级到 2008 的变化却是更细微的。下面列出了 SQL Server 2008 数据复制功能的新特性。

1、极大地提高了快照和事务复制命令执行效率；

2、为创建和修改 Peer-to-Peer（P2P）网络拓扑提供了新界面；

3、对于 Peer-to-peer 点对点数据复制功能，新版本中支持冲突检测，而且无需离线即可复制 schema 的变化；

4、深度复制与数据库镜像和 log shipping（日志传送）整合；

5、为开发人员提供了数据库同步类库（Sync Services），开发人员无需掌握 DBA 管理员知识即可使用；

6、调整了复制监视器功能。

下面我们详细看一下这些新特性。

快照和事务复制命令执行效率的提高

快照技术是指从发布者向订阅者复制数据，数据库对象和可以重建表，存储过程，函数，视图等的元数据。

微软博客上的这个帖子有一些关于提高可扩展性的内容，但是最令人感兴趣的部分是下面这段话：

“结果绝对令人瞠目。Pull 方式的复制在 Yukon 平台（Windows Server 2003）的 SQL Server 2005 上运行需要 223 分钟，而在采用 Katmai 指令技术的 Longhorrn Server 平台（Windows 2008 Server）上只需要 2 分钟”.一份 11.3GB 大的快照在大约 23 分钟内被发送到 2000 英里之外（几乎每分钟 500MB 的速率）。这是按字节算的，而不是按位算的。可以讨论‘better together’了。

除了快照执行效率的提升，运行在 Windows Server 2008 上的 SQL Server 2008 可以利用 TCP/IP 网络协议栈改善复制命令的延迟时间。Windows Server 2008 对于 TCP/IP 网

络协议栈的改进使得我们可以自行调整 TCP/IP 包的参数（某些情况下是自动调整的）。涉及的具体参数有：

1、接收 Window Auto-Tuning

2、发送缓冲自动扩展

3、接收 Compound TCP

按照 MSDN 上这份白皮书“geo-replication performance gains”，微软的 Engineering Operations 团队在 pull 类型数据复制上，达到了令人难以置信的性能提升：11，345.13%。

为创建和修改 Peer-to-Peer（P2P）拓扑提供了新界面

Peer-to-peer（P2P）数据复制是事务复制的一种变体，多个节点会给网络拓扑中的其他节点发布变化内容。如果一个节点与网络断开了连接，它可以继续工作，网络拓扑中的其他节点会互相执行复制。当该断开的节点再次连接到网络中来时，它会从网络拓扑中的其他节点接收到离线时产生的所有变更。新界面可以复制离线时产生的变化。这种复制类型是网状拓扑结构，而不是发布/订阅方式的拓扑。发布订阅的拓扑中，发布者为事务清理空间并决定变更发生的地方。

图 1 展示了新版本中的 Peer-to-peer 网络拓扑界面，图 2 显示了怎样给网络拓扑中添加一个节点。

图 1：peer-to-peer（P2P）网络拓扑向导

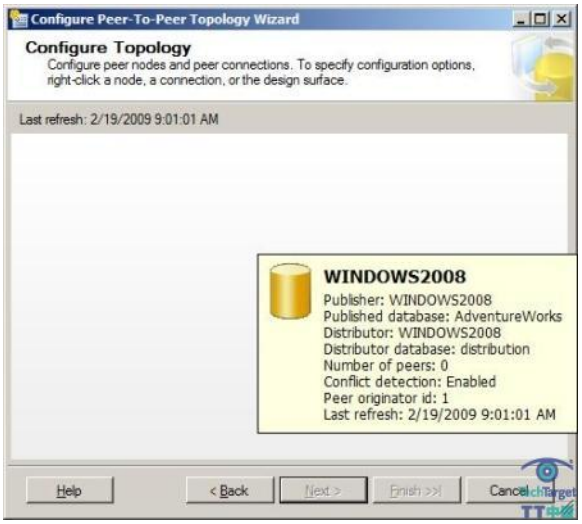
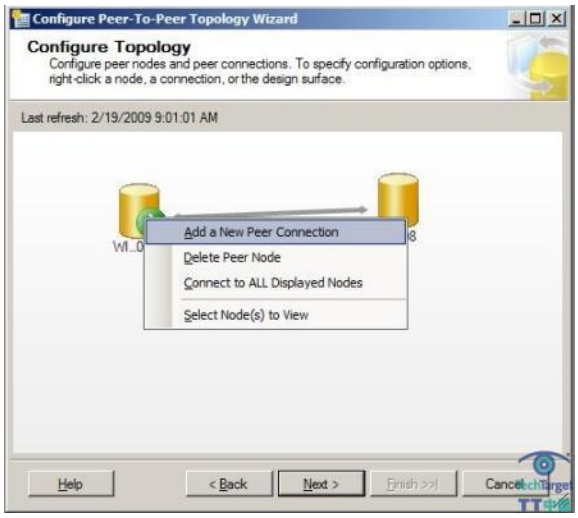


图 2：给 peer-to-peer 网络拓扑添加新节点



（作者：Hilary Cotter 译者：冯昀晖 来源：TT 中国）

原文标题：SQL Server 2008 数据复制新特性及其带来的价值（上）

链接：http://www.searchdatabase.com.cn/showcontent_21891.htm

SQL Server 2008 数据复制新特性及其带来的价值（下）

Peer-to-peer（P2P）数据复制增强功能

在 SQL Server 2005 中，修改数据复制的网络拓扑或修改正在复制的表是个挺麻烦的过程。DBA 们不得不先停止所有节点上的活动，检查确认所有的变更已经被复制，然后再应用变更;一旦再次确认所有变更已经被复制到所有节点之后，才在最后允许用户回来使用。

在 SQL Server 2008 中，任何时候你都可以执行更改。你可以在不限制用户正常访问系统的情况下，添加或删除节点，修改数据库 schema。SQL Server 2008 还支持冲突检测，该功能默认状态是启用的。当你试图更改已经在另一个节点上删除的一行数据时，或者在不同节点上更新同一行时，或者同时在不同节点上赋值相同的主键时，冲突就会出现。在 SQL Server 2005 中，这会引起分发代理程序运行失败，你不得不手工修复受影响节点的问题。如果存在大量冲突的话，这些工作做起来会非常复杂和费力。

你可以禁用 SQL Server 2008 中的冲突检测功能，这样分发代理程序会忽略出现的冲突。微软推荐你把数据分区，尽可能避免冲突。

深度复制与数据库镜像和 log shipping（日志传送）整合。

SQL Server 之前的版本要求 DBA 们对基于日志传输的发布者数据库重新初始化订阅。而在 SQL Server 2008 中，对镜像数据库或备用服务器执行故障切换，并让数据继续向订阅者复制是可能的。你可以在白皮书“providing high availability using database mirroring”中找到详细的指导。

Sync Services 同步服务类库

Sync Services（同步服务）是一个.NET 类库，你可以使用它在各种不同类型的数据源之间同步变化。同步服务类库 Sync Services 是由微软开发 merge replication（合并复制）功能的同一团队设计开发的。Sync Services 的目标是创建轻量级的类来执行合并复制的基本功能，无需维护和管理。下面的链接解释了 Sync Services 和 merge replication（合并复制）之间的区别，基本上归纳了你能做的范围。换句话说，开发人员应该使用 Sync Services，而 DBA 们应该使用 Merge replication（数据复制）功能。需要指出的是：Merge replication 比 Sync Services 有更多的功能特性，它是向导驱动的。

复制监视功能的改进

在 SQL Server 2008 中，Replication Monitor 有几个细微的变化。我们看看它与 SQL Server 2005 中 Replication Monitor 的比较：

SQL Server 2005 中，Replication Monitor 每个发布器有三个 tab 页：

Publication（发布）

Subscription Watch List （订阅监视列表）

Common Jobs（通用任务）

SQL Server 2008 中，Replication Monitor 每个发布器中有三个 tab 页：

Publication（发布）

Subscription Watch List （订阅监视列表）

Agents（代理）

SQL Server 2005 中，Common Jobs Tab 页功能包括：

Replication Agents Checkup （复制代理审查）

Reinitialize Subscriptions Having Data Validation Failures （数据校验失败时重新初始化）

Distribution Clean Up （分发清理）

Agent History Clean Up （代理历史清理）

Replication Monitoring Refresher （复制监视刷新）

Expired Subscription Cleanup（过期订阅清理）

SQL Server 2008 复制监视代理 Tab 也有一个下拉列表，其中显示了 snapshot agent（快照代理），log reader agent（日志阅读器代理），queue reader agent（队列读取器代理）或者所有维护任务。其中维护任务与 SQL Server 2005 中显示的 Common Job Tab 页的功能一样。

如果你深入使用 SQL Server 2005 复制监视器 publication，你会看到两个或三个 tab 页（取决于你的复制类型）如下：

All Subscriptions （所有订阅）

Tracer Tokens （只有事务发布功能有此 tab 页）

Warnings and Agents（警告语代理）

SQL Server 2008 有三个或四个 tab 页

All Subscriptions （所有订阅）

Tracer Tokens （只有事务发布功能有此 tab 页）

Warnings （警告）

Agents（代理）

以上列出了 SQL Server 2008 数据复制功能的主要更新特性，其中最令人感兴趣的要数在 windows 2008 上运行 SQL Server 的 snapshot 和事务执行效率的提升了。

（作者：Hilary Cotter 译者：冯昀晖 来源：TT 中国）

原文标题：SQL Server 2008 数据复制新特性及其带来的价值（下）

链接：http://www.searchdatabase.com.cn/showcontent_21896.htm

SQL Server 2005 的复制存储过程选项

如果 SQL Server 上有正确的过程和技术，那么在数据库之间复制存储过程可以是一项简单的任务。其中将包括本机的复制功能或者日志传输。在没有这些功能的情况下，我们仍然可以使用手动过程来实现相同的操作。事实上，在特定的环境和情况下手动的方法可能效果会更好一些。

内置复制功能

通过使用 SQL Server 2005 复制，存储过程定义是以一些不同的方式来复制的。快照（Snapshot）复制选项能够移动所有对象和数据，因此，数据库对象的定义能在发布者（Publisher）和订阅者（Subscriber）之间复制。当创建发布（Publication）和定义一个物件（Article）时，事务和整合复制都有相同的功能来移动存储过程的定义。这是通过指定 sp_addarticle 系统存储过程的@type 变量的正确参数来实现的。这些变量的选项是：

- o proc schema only —— 支持存储过程模式
- o proc exec——支持存储过程执行
- o serializable proc exec——支持在一个序列化事务中的存储过程执行
- o func schema only —— 支持一个只用于模式的用户定义方法

其中值得一提的一项是 sp_addarticle 的@type 变量的功能，它是用来在订阅者上复制存储过程的执行的。我们可以通过 proc exec 或序列化 proc exec 参数来实现相同的操作。因此，如果我们有一些主要的事务是通过存储过程来执行的，那么，这就是一个通过存储执行复制数据而非直接复制数据的选项。关于更多这个选项的其它信息，可以参考在事务性复制中发布存储过程执行和如何：在事务性发布（SQL Server Management Studio）中发布存储过程的执行。

日志传输

使用日志传输，通过发布一个全面和事务的日志备份，生产服务器上的所有存储过程代码都会被复制到备份服务器上。这样就确保，即使代码在生产环境发生变化，它也将会与在备份服务器上的代码相同。它的价值在于，没有完成任何其它的任务的情况下——包

括安装和监控备份和恢复——不再需要其它的操作来确保代码在环境之间正确的复制。这对于灾难恢复和高可用需求是非常有益的，但是它可能并不支持我们的应用。

手动部署

当编译代码或者使用 SQL Server 2005 Management Studio 的本机功能时，我们可以通过修改脚本来在多服务器上执行相同的代码。这样，我们就能够将代码修改进行分组，并通过连接到正确的 SQL Server 数据库手动复制代码。我们还可以整合一个版本控制系统来扩展这一功能。其中一个这样的系统是 Visual SourceSafe，当在 QA 环境中测试和对产品进行部署时，它支持代码在开发时的编译和在版本控制系统中的取出和提交。

自动部署和同步

最后一个选项是利用市场上某些第三方产品以便自动化存储过程的部署过程。有些工具支持自动的方式在并行的服务器之间进行打包和部署代码。有些工具可以同步化环境之间的代码。我们可以从供应商中了解这些工具。

不要忘记过程和技术

本文中所列出的技巧提供了一个在 SQL Server 环境中复制存储过程的合理技术方法。可惜的是，没有一个存储过程是孤立的，我们也无法将用户和过程分离开来。事实上，代码修改要求前端、中间层和其他 DML\DDL 的修改。此外，在开发、测试和生产环境之间，变更管理过程必须确定一个固有的过程。一定要为正确解决方案权衡好技术、人员和过程。

(作者：Jeremy Kadlec 译者：曾少宁 来源：TT 中国)

原文标题：SQL Server 2005 的复制存储过程选项

链接：http://www.searchdatabase.com.cn/showcontent_21747.htm

如何在升级到 SQL Server 2005 时复制数据库？

问：我打算将大概 300 个数据库从 SQL Server 2000 32-bit 迁移到 SQL Server 2005 64-bi。我已经尝试了使用“复制数据库”的功能但是失败了。此外我还需要移开所有登录。希望你能给我提一些建议。

答：我发现数据库复制向导很慢，在效率方面要比迁移数据的其他方法差一些。如果你需要在一系列服务器之间进行少量数据的快速迁移，那它就是最佳解决方法，你也不必花时间对数据库进行备份或者卸载数据库。

如果你需要迁移 300 个数据库，我觉得你会发现最快的方法就是在 SQL Server 2000 实例上使用 sp_detach_db 卸载每个数据库并且在 SQL Server 2005 实例上重建数据库。这样做还有利于保护你的数据库用户。

但是即使用这种方法你也仍然要登录 SQL Server 2005 实例。你可以查看 Microsoft Knowledge 里标题为《如何在 SQL Server 实例之间迁移注册程序和密码》这篇文章。这篇文章包括你所需要的脚本。

注意：如果你采用这种方法，你就需要重测数据库注册程序。你可以 sp_change_users_login 存储程序来完成。

(作者：Adam Machanic 译者：April 来源：TT 中国)

原文标题：如何在升级到 SQL Server 2005 时复制数据库？

链接：http://www.searchdatabase.com.cn/showcontent_14550.htm