SQL Server 服务代理指南

SQL Server 服务代理指南

在 Denny 先生的博客中，SQL Server 专家 Denny Cherry 讨论了 SQL Server 功能基础。其中一个话题主要是围绕 SQL Server 服务代理，和 MSMQ 有着相同的基础概念的消息队列（message queue）系统。无论你是一名新手还是一名专家，本篇指南都会向你提供非常有用的、有关如何建立和调整 SQL Server 这一新补充方面的信息。

SQL Server 服务代理概述

SQL Server 服务代理能够保证消息发送顺序，总是对按顺序接收的消息做出相同的处理过程。消息可以按要求进行处理（你可以编写软件来按时间查询服务代理）或者自动地队列激活。

- **基础知识：SQL Server 服务代理**

SQL Server 服务代理消息类型

消息类型是服务代理相关的第一个对象。消息类型定义了消息名称、以及服务代理必须在消息发送之前要做的验证。

- **SQL Server 服务代理的消息类型**

SQL Server 服务代理的契约

每一个服务代理会话都需要一个契约（Contract）。这个契约定义了哪个或哪些类型的消息会被使用，以及谁（发送者和接收者）可以使用这些类型的消息。契约是在会话创建时指定的。

❖ SQL Server 服务代理的契约

SQL Server 服务代理的队列

服务代理队列是消息被发送后存储的地方，但队列中的消息是还没有被处理的。队列有点像数据库表一样，你可以用来插入数据（发送消息）以及查看表中的数据（接收消息）。

❖ SQL Server 服务代理的队列

SQL Server 服务代理的服务

服务是用于将契约绑定到队列上的。它们也用于数据库到数据库，或者服务器到服务器之间的消息路由（我们将在后面讨论路由）。

SQL Server 服务代理的服务

SQL Server 服务代理的路由

路由只是在从一个服务器向另一个服务器发送服务代理消息时才会用到。它们定义了发送消息所需要的 SQL Server 和 SQL Server 将要连接的 TCP 端口。如果你是向一个镜像数据库发送消息，你也可以像指定主数据库一样指定该镜像数据库。

❖ SQL Server 服务代理的路由

SQL Server 服务代理的 SEND 命令

SEND 命令是在 SQL Server 2005 版本引入的。它是一种类似于 INSERT 的命令，只是 SEND 只用在 SQL Server 服务代理中。在服务代理中，你是将数据填入一个队列，而不是

数据库表中。你并不是把命令插入到队列中，而是向队列发送一个消息，这就像你向其他人发送一个电子邮件一样。

- SQL Server 服务代理的 SEND 命令

SQL Server 服务代理的 RECEIVE 命令

RECEIVE 命令用于从队列中获取消息，这样你就可以对 message_body 字段中的数据进行处理。从本质上来说，RECEIVE 命令在很多情况下与 SELECT 语句的使用方法很相似。

- SQL Server 服务代理的 RECEIVE 命令

SQL Server 服务代理的消息已删除错误

我经常遇到有关消息删除的问题，可能其他人也会遇到了同样的问题。你仍然能够在 sys.conversation_endpoints DMV 中看到该会话仍处于“会话中（CONVERSING）”状态，而不是“关闭（CLOSED）”状态。

- SQL Server 服务代理的消息已删除错误

改进 SQL Server 服务代理的性能

在高负载的服务代理环境中，通过重用服务代理会话能够大大提升服务代理的性能。为每一个消息创建和关闭会话需要的开销大约是 4X，而对于接收消息的性能提升大约能达到 10X。Remus Rusanu 在他发布在 Reusing Conversations 的博客上，谈到这个估值可能更大，并且还提出了一种重用会话的解决方案。

- 改进 SQL Server 服务代理的性能

监控 SQL Server 服务代理的运行状况

在处理 SQL Server 服务代理问题时，我需要一个快速简单的方式来查看服务代理的积压事务。所以我选择使用下面的这样一个小查询，它非常有效……

- 监控 SQL Server 服务代理的运行状况

基础知识：SQL Server 服务代理

SQL Server 服务代理是 SQL Server 的一个非常好的新补充。对于曾经使用过 Microsoft Message Queue (MSMQ)的人来说服务代理是很容易理解的，因为它他们概念上是相同的。消息都是从一个地方发送到另一个地方，可以是在一个数据库内部，可以是从一个数据库到另一相数据库，或者是从一个服务器到另一个服务器。

虽然服务代理的配置是一件相当复杂的任务，但在服务代理创建好后，它是一个能够处理大量消息负载的非常稳定的系统。

服务代理能够保证消息发送顺序，它总是对按顺序接收的消息做出相同的处理过程。消息可以按要求进行处理（你可以编写软件来按时间查询服务代理）或者自动地队列激活。

回顾一下关于各种服务代理对象的信息，以及如何对它们进行配置。

（作者：SearchSQLServer.com 译者：曾少宁/陈柳 来源：TT 中国）

SQL Server 服务代理的消息类型

消息类型是服务代理相关的第一个对象。消息类型定义了消息名称，以及服务代理必须在消息发送之前要做的验证。如果消息是被发送到另一个数据库（在同一个服务器或不同服务器上的），那么消息类型必须存于两个数据库中。

验证可能是下面四种方式之一：

1. None (My personal favorate especially for sample code)
2. Empty
3. Well_Formed_XML
4. Valid_XML With Schema Collection

第 1 和 2 种非常简单。None=不作任何验证。Empty=消息必须是空的。Well_Formed_XML 要求 XML 必须是一个有效的 XML 文档。第 4 种不仅要求 XML 是有效的，而且它要符合已有的 XML 模式集（加载到 SQL Server 的 XSD）。

创建消息类型的语法也是非常简单的：

```
CREATE MESSAGE TYPE [MessageTypeName]
VALIDATION = NONE
```

就是这样。如果你需要修改 XML 模式，使用 AUTHORIZATION 子句去设置对象所有者。

使用 NONE 的验证时，则由接收端代码或应用负责验证消息中数据是否存在和有效。

（作者：SearchSQLServer.com 译者：曾少宁/陈柳 来源：TT 中国）

SQL Server 服务代理的契约

每一个服务代理会话都需要一个契约（Contract）。这个契约定义了哪个或哪些类型的消息会被使用，以及谁（发送者和接收者）可以使用这些类型的消息。契约是在会话创建时指定的。

创建契约的语法也是非常简单的。

```
CREATE CONTRACT [ContractName]
(MessageType SENT BY ANY)
```

其中 SEND BY 部分可以在每一个定义在契约中的消息类型中多次使用。但它至少应该出现一次，因为你必须给契约指定一个消息类型。你可以单独或同时（使用 ANY 关键字）指定 INITIATOR 和 TARGET 的消息类型。

下面是一个更复杂的创建契约的例子：

```
CREATE CONTRACT [ContractName]
(MessageType_I SENT BY INITIATOR,
 MessageType_T SENT BY TARGET)
```

（作者：SearchSQLServer.com 译者：曾少宁/陈柳 来源：TT 中国）

SQL Server 服务代理的队列

服务代理队列是消息被发送后存储的地方，但队列中的消息是还没有被处理的。队列有点像数据库表一样，你可以用来插入数据（发送消息）以及查看表中的数据（接收消息）。数据库表是有触发器的，而队列则有激活的过程（Procedure）。如果队列定义了一段激活的过程，当消息到达队列后，队列会执行该过程。执行过程中，我们不需要向过程传递数据，该过程会自己执行 RECEIVE 来获取数据（如果确实需要这样做的话）。

配置一个激活的过程有些复杂，因为你必须使用队列来创建该过程，并且让过程自己设置为激活的。所以我喜欢的方式的是先创建队列，然后是过程，再让队列去激活过程。使用这种方法，我可以马上发送消息给队列，而不需要担心会出错，因为此时过程并不存在。

与数据库表不同，你不需要定义列表结构。所有数据都是存储在 message_body 域中，所以如果需要发送多个值，我推荐你使用 XML 文档来发送数据。创建列表的语法是非常简单的：

```
CREATE QUEUE [QueueName]
```

因为队列中的消息已经定义了队列中数据应该存储的文件组。如果你需要在消息处理后仍然保存该消息，那么你可以用关键字 RETENTION 将其设置为 ON。默认情况下队列在创建后状态就是激活的。如果你不允许向队列发送消息，你可以用关键字 STATUS 将队列设置为 OFF。

（作者：SearchSQLServer.com 译者：曾少宁/陈柳 来源：TT 中国）

SQL Server 服务代理的服务

服务是用于将契约绑定到队列上的。它们也用于数据库到数据库，或者服务器到服务器之间的消息路由（我们将在后面讨论路由）。跟 SQL Server 中其它大多数的对象不同，不管你在数据库设置了什么样的校对（collation），服务的名称都是大小写敏感的。如果使用服务器到服务器队列，服务的名称必须是相同的，这包括 SQL Server 使用服务名哈希值来定位服务的情况。（如果你不使用路由，服务名称实际上不是大小写敏感的，但是如果你可能会在将来某个时间开始使用消息路由，所以最好一开始就正确区分大小号。）

创建服务的语法是非常简单的。

```
CREATE SERVICE ServiceName
ON QUEUE QueueName (ContractName)
```

如果服务只是发起者，那么契约名可以不用指定。如果你想创建多用户契约，你可以通过使用逗号分隔的契约列表名指定额外的契约。

```
CREATE SERVICE ServiceName
ON QUEUE QueueName;
CREATE SERVICE ServiceName
ON QUEUE QueueName (ContractName1,  ContractName2);
```

（作者：SearchSQLServer.com 译者：曾少宁/陈柳 来源：TT 中国）

SQL Server 服务代理的路由

路由只是在从一个服务器向另一个服务器发送服务代理消息时才会用到。它们定义了发送消息所需要的 SQL Server 和 SQL Server 将要连接的 TCP 端口。如果你是向一个镜像数据库发送消息，你也可以像指定主数据库一样指定该镜像数据库。

如果你需要建立 3 个或更多的服务器链，并从一个服务器向另一个服务器转发消息；如果你需要让消息穿越防火墙，但是源服务器和目标服务是不允许在防火墙两端进行通信；路由是非常好用的。其中唯一的前提条件是每一个会话中有一个服务器运行着 SQL Server。换句话说，两个 SQL Express 实例是不能互相直接发送消息的。这些消息必须通过 Workgroup、Standard 或 Enterprise 转发。

在你创建一个服务代理路由之前，你必须有一个远程计算机终端。为此，我们假定服务代理终端是创建在 1234 端口上。我们本地服务器是 SQL1，而远程服务器是 SQL2。你需要知道的另一个信息是 SQL2 上的服务代理实例的 GUID。它就在 server2 的 sys.databases DMV 的 service_broker_guid 字段上（SQL 2005 中是从右数第 5 个字段）。如果 GUID 全为 0，那么你需要使用 ALTER DATABASE 命令也激活服务代理。

CREATE ROUTE 命令的语法非常简单。

```
CREATE ROUTE RouteName
WITH SERVICE_NAME = ServiceName,
BROKER_INSTANCE = ae8505fa-b84d-4503-b91f-
3252825ccf09,  /*Use your GUID here*/
ADDRESS=TCP://SQL2:1234
```

如果你想使用数据库镜像，可以将 MIRROR_ADDRESS 设为镜像的名称和端口号。通过这种方法，发送服务器在崩溃后仍然能够继续发送消息。

如果你需要路由在某个日期后失效，比如你发送数据给合作伙伴，但希望在合同结束后能自动停止向对方继续发送消息，你可以添加 LIFETIME 参数，指定路由过期的秒数。如果没有指定 LIFETIME，该路由会长期有效。

（作者：SearchSQLServer.com 译者：曾少宁/陈柳 来源：TT 中国）

SQL Server 服务代理的 SEND 命令

SEND 命令是在 SQL Server 2005 版本引入的。它是一种类似于 INSERT 的命令，只是 SEND 只用在 SQL Server 服务代理中。在服务代理中，你是将数据填入一个队列，而不是数据库表中。你并不是把命令插入到队列中，而是向队列发送一个消息，这就像你向其他人发送一个电子邮件一样。服务代理也使用相同的做法（更准确地说，它们并不完全相同，因为电子邮件是每一个人都能接收的）。

当你使用 SEND 命令时，你必须先使用 BEGIN DIALOG 命令。所以在你发送消息之前，你必须先启动一个对话框，它再启动一个会话，然后使用服务去完成实际的逻辑操作。这是通过 BEGIN DIALOG 命令来完成的，如下的示例代码。

在你获得了@dialog_handle 变量中的对话句柄后，你就可以使用这个句柄发送实际的消息到之前你启动对话框时开始的会话。在我们下面的例子中，我们只是发送 sys.tables DMV 的内容到队列中：

```
DECLARE @xml AS XML
DECLARE @dialog_handle AS uniqueidentifier SET @xml = (SELECT * FROM sys.table
s FOR XML AUTO)BEGIN DIALOG @dialog_handle
FROM SERVICE [tcp://codecamp/AW/sample_send_service]
TO SERVICE tcp://codecamp/AW/sample_receive_service
ON CONTRACT [tcp://codecamp/AW/sample_contract];SEND ON CONVERSATION @dialog_h
andle
 MESSAGE TYPE [tcp://codecamp/AW/sample_messagetype]
 (@xml);
```

还不太清楚，是吗？我的博客后面很快就会有关于 RECEIVE 命令的内容更新，你可以用它来读取消息内容。并且在更新中我会给一个示例代码链接，那些代码是我在文章中多次使用的，它们应该能帮助你融会贯通这些内容。

（作者：SearchSQLServer.com 译者：曾少宁/陈柳 来源：TT 中国）

SQL Server 服务代理的 RECEIVE 命令

RECEIVE 命令是本系列文章讨论的关于服务代理的最后一个方面内容。RECEIVE 命令用于从队列中获取消息，这样你就可以对 message_body 字段中的数据进行处理。RECEIVE 命令在很多情况下与 SELECT 语句的使用方法很相似：它也有一个 FROM 子句和一个 WHERE 子句（这并不常用，但需要时是可以使用的）。在你接收到消息后，通常你执行 END CONVERSATION 命令。这将会告诉远程计算机你已经接收到了消息而且不应该再有消息在这个会话中发送了。我通常在每一个个会话中只发送一个消息，这样我就不需要有多余的逻辑代码去检查这是不是会话的最后一条消息。

这个命令的基本语法很简单。

```
    DECLARE @handle UNIQUEIDENTIFIER
DECLARE @message_body XML
RECEIVE TOP (1) @handle=conversation_handle,  @message_body=cast(message_body
as XML)
FROM [QueueName]END CONVERSATION @handle
```

它要求你将消息内容强制转换成 XML 格式，因为它是以二进制数据格式传输的。（如果你发送的是一个值或二进制 blob 数据，你就需要修改转换格式。）这样在你接收到消息后，你不需要再执行一个命令去队列中删除该消息。RECEIVE 命令能在一个命令中同时处理 SELECT 和 DELETE 操作。如果你不能够理解，这里有我用于处理服务代理会话的一个示例源代码。ZIP 文件中包括了示例源代码和关于我这个博客内容的演示文稿。如果你有任何关于示例源代码的问题，你可以在这里提出你的问题，这样大家都能看到我的回复。

我可能会暂时停止“Back To Basics”博客的更新，但我还没有完全决定这样做，所以如果你还能看到我的更新，这表示我已经决定继续这个系列博客的更新。相反，我可能

会暂停一小段时间。

我保证。

Denny

(作者：SearchSQLServer.com 译者：曾少宁/陈柳 来源：TT 中国)

SQL Server 服务代理的消息已删除错误

我曾经看到这样一个关于服务代理的问题，也许其他人也看到过的：消息被发送到服务代理，接着进行正常的处理，然后它们会从消息队列中删除。然而你仍然能够在 sys.conversation_endpoints DMV 中看到该会话仍处于“会话中（CONVERSING）”状态，而不是“关闭（CLOSED）”状态。而在 sys.transmission_queue 并没有相应的记录，这是非常奇怪的。同时，当我们使用 SQL Server Profiler 检查时也不会发现错误。

很明显这是一个已知问题。但奇怪的是在我的系统中，它只发生在数据库的一个队列里。

目前唯一的措施就是对该会话做一个中止并清除（END CONVERSATION WITH CLEANUP）的操作。我已经写了一个脚本用于清除会话。脚本中我专门做了判断，它只清除有问题的会话中的消息，这些会话是当前不在队列中的（这个队列不是自动处理的，有一个服务会每隔 30 秒检查一次队列，所以这样就能够保留队列中一些我不想删除的有效消息）。

```
declare @i int
set @i = 1
while @i <> 10000
begin
  declare @conversation_handle uniqueidentifier
  declare cur CURSOR for
  SELECT TOP (1000) conversation_handle
  FROM sys.conversation_endpoints
```

```
  WHERE NOT EXISTS (SELECT *
  FROM [tcp://AWT/Sonar/Q_ObjectDelete] a
  WHERE a.conversation_handle = sys.conversation_endpoints.conversation_handle
)
  AND sys.conversation_endpoints.far_service = tcp://AWT/Sonar/Svc_ObjectDelet
e
  AND sys.conversation_endpoints.state <> CD
  open cur
  fetch next from cur into @conversation_handle
  while @@fetch_status = 0
  begin
  end conversation @conversation_handle with cleanup
  fetch next from cur into @conversation_handle
  end
  close cur
  deallocate curset @i = @i + 1
endI run this every hour to clean up the bogus records in the sys.conversation
_endpoints DMV.
```

如果不清理 sys.conversation_endpoints DMV，tempdb 会慢慢被填满，并在 sp_spaceused 显示 tempdb 为空时抛出空间耗尽消息。这跟我之前在另一篇文章所说的一样。

（作者：SearchSQLServer.com 译者：曾少宁/陈柳 来源：TT 中国）

改进 SQL Server 服务代理的性能

之前我与一个 Microsoft Support Engineer 交谈时，他提到在一些像我们这样的高负载的服务代理环境中，通过重用服务代理会话能够大大提升服务代理的性能。

为每一个消息创建和关闭会话需要的开销大约是 4X，而对于接收消息的性能提升大约能达到 10X。Remus Rusanu 在他发布在 Reusing Conversations 的博客上，谈到这个估值可能更大，并且还提出了一种重用会话的解决方案。

我喜欢 Remuss 的解决方案，但其中有一个问题：我不愿意为每一个 SPID 创建不同的会话。如果我使用这种方法，就会有大量的会话打开，这样我还需要去关闭它们。在应用中，许多问题都可能触发一个服务代理消息的发送。而且通常情况下都会有大量的线程在同一时间对数据库进行访问。这就要求我们在实际环境使用 Remuss 的方法之前要对它进行优化，使之更灵活。我的想法是为我们应用中的每一个进程创建一个会话。并且在一定周期内，有一些会话会被关闭，也有些新的会话被创建。

首先我先介绍一些关于安装服务代理的背景知识。我已经有 2 个队列和 2 个服务。每一个服务都在它自己的队列中。同时我有一个专门的消息类型，它用于将消息从源发送到目标位置。

- 我的消息类型是 MT_ObjectDelete，
- 我的契约：CT_ObjectDelete，
- 我的队列：Q_ObjectDelete_Source 和 Q_ObjectDelete_Destination，
- 我的服务：Svc_ObjectDelete_Source 和 Svc_ObjectDelete_Destination，

另外我的另一个消息类型是 MT_ConversationSwitch，它也是契约 CT_ObjectDelete 的一部分。

为了存储我将生成的会话句柄，我会使用现有一个保存各种配置信息的表 Setting。我已经添加了一些新的会话终端记录到表中。其中在 Setting 表中的查询名称是 SSB_Session_Delete。我使用 GetSystemSettingValue 方法来查询 Setting 表得到配置值。发送数据的存储过程是这样的：

```
CREATE PROCEDURE [dbo].[SendDelete]
@FileName varchar(1024)
AS
DECLARE @DialogHandle uniqueidentifier,
@msg XML,
@date nvarchar(100)
BEGIN
IF @msg IS NOT NULL
BEGIN
    SET @DialogHandle = cast(dbo.GetSystemSettingValue('SSB_Session_Delete') a
s uniqueidentifier)
    IF CAST(RAND()*100 AS INT) = 0 OR @DialogHandle IS NULL
BEGIN
IF @DialogHandle IS NOT NULL
SEND ON CONVERSATION @DialogHandle
MESSAGE TYPE [MT_ConversationSwitch]

BEGIN DIALOG CONVERSATION @dialogHandle
FROM SERVICE [Svc_ObjectDelete_Source]
TO SERVICE Svc_ObjectDelete_Destination
ON CONTRACT [CT_ObjectDelete_Multi];
        UPDATE Setting
SET DefaultValue = @DialogHandle
```

```
WHERE SettingName = SSB_Session_Delete;
END;
SEND ON CONVERSATION @dialogHandle
MESSAGE TYPE [MT_ObjectDelete]
(@msg)

IF @DialogHandle <> cast(dbo.GetSystemSettingValue(SSB_Session_Delete) as uniq
ueidentifier)
SEND ON CONVERSATION @DialogHandle
MESSAGE TYPE [MT_ConversationSwitch]
END
END
GO
```

你可以从代码中看到，我通过查询获取当前的会话句柄。如果没有找到句柄，我会创建一个新句柄并将句柄值保存到我的 Setting 表中。如果 CAST(RAND()*100 AS INT)返回 0，我会使用现在会话中的 MT_ConversationSwitch 消息类型发送一条消息，它将触发接收程序的逻辑，终止该会话。在我终止该会话后，我再创建新的会话并将它存储在数据库中。

然后我在该会话句柄上发送一条消息。接着我会再次检查 Setting 表，以确保我使用的值与表中的值是相匹配的。如果不匹配，我会认为另一个线程已经在相同的代码块中运行，然后关闭该会话，因为我不希望该会话继续存在。

我的接收程序非常简单：

```
CREATE PROCEDURE [dbo].[util_ProcessDelMonitorData]
```

```
@MsgToRead INT = 1000
AS
DECLARE @message_type_name sysname;
DECLARE @message_body VARBINARY(max)
DECLARE @msgTable TABLE
(
message_body VARBINARY(max),
[conversation_handle] UNIQUEIDENTIFIER
);
BEGIN
DECLARE @conversation_handle UNIQUEIDENTIFIER
WHILE 1=1
BEGIN
WAITFOR (RECEIVE TOP (1) @Message_Body = message_body,
@conversation_handle = [conversation_handle],
@message_type_name = message_type_name
FROM [Q_ObjectDelete_Destination]),  TIMEOUT 1000
    IF @conversation_handle IS NULL
BEGIN
break
END
IF @message_type_name = MT_ConversationSwitch
END CONVERSATION @conversation_handle
    IF @message_body IS NOT NULL
BEGIN
INSERT INTO @msgTable
(message_body,  [conversation_handle])
```

```
values
(@message_body,  @conversation_handle)
END
    SET @conversation_handle = NULL
SET @message_type_name = NULL
END
/*Business Logic happens here*/
END
GO
```

你可以看到，我会对于第一条消息做标准的接收。如果消息没有发现值，我会转而处理我之前存储在@msgTable 表变量中的数据。

如果消息类型 message_type_name 是 MT_ConversationSwitch，我会对该会话做一个中止会话（END CONVERSATION）的操作。

如果@message_body 变量中有数据，我会将该值存储到表变量中，给后面的业务逻辑进行处理。

另外在发送队列中还有一个激活的程序，它只是对它接收的每一个消息做简单的中止会话（END CONVERSATION）操作。

我真心希望这个例子能对你有所帮助。如果你对此有任何问题，请提交你的问题，我将尽快地回复你。

（作者：SearchSQLServer.com 译者：曾少宁/陈柳 来源：TT 中国）

监控 SQL Server 服务代理的运行状况

在处理 SQL Server 服务代理问题时，我需要一个快速简单的方式来查看服务代理的积压事务。所以我选择使用下面的这样一个小查询，它非常有效：

```
select far_service,  state_desc,  count(*) messages
from sys.conversation_endpoints
group by state_desc,  far_service
ORDER BY far_service,  state_desc
```

希望这对你有所帮助。

Denny

（作者：SearchSQLServer.com 译者：曾少宁/陈柳 来源：TT 中国）