SQL Server 2005

Administration und Programmierung

ADDISON-WESLEY

7 Notification Services und Service Broker

Im letzten Kapitel habe ich erläutert, wie Sie mithilfe der Replikation Daten von einem System an ein anderes senden können. Diese Erörterung stand im Zusammenhang mit Hochverfügbarkeit, aber Sie können die Replikation selbstverständlich auch für viele andere Bereiche einsetzen, die einen Austausch von Daten erfordern.

In diesem Kapitel befassen wir uns mit einem gezielteren Ansatz für die Verteilung von Daten. Mit Notification Services können Sie ausgehend von Bedingungen in Ihrer Datenbank bestimmte Datagramme an Benutzer senden, zum Beispiel eine Benachrichtigung über die Änderung eines Aktienkurses oder eine Warnung, dass ein Rechner gerade offline gegangen ist. Jedes Ereignis, das Sie in Ihrer Datenbank nachverfolgen, steht für die Verwendung in Notification Services zur Verfügung. Anders als bei der Replikation können Sie mit Notification Services Daten an eine E-Mail-Adresse, ein Mobiltelefon oder über eine SMS (*Simple Messaging Service*) an ein beliebiges Gerät senden, das diesem Standard entspricht.

Notification Services lässt sich auf mehrere Arten einsetzen. Sie können das Programm hauptsächlich als Aufgabe ansehen, für die der Datenbankadministrator zuständig ist, oder es für Ihre Entwickler aktivieren. In diesem Kapitel konzentriere ich mich auf die Administratorseite, worüber Sie aber die vielfältigen Anwendungsmöglichkeiten nicht vergessen sollten.

Sie können SQL Server 2005 als Teil einer belastbaren dienstorientierten Architektur (*Service-Oriented Architecture, SOA*) nutzen, die eine neue Art der Erstellung von Anwendungen ermöglicht, bei der die Auslastung mithilfe von gegenseitigen Nachrichten auf mehrere Server verteilt wird. SQL Server 2005 stellt Service Broker als programmierbares Objekt bereit, das Sie als Mechanismus zum Speichern und Weiterleiten verwenden können. Möglicherweise schreiben Sie keine Programme dieser Art, sind aber als Datenbankadministrator dafür zuständig, den betreffenden Teil des Frameworks für eine dienstorientierte Architektur einzurichten und zu verwalten.

Im zweiten Teil des Kapitels geht es darum, wie Sie mithilfe von Service Broker zur Entwicklung und Verwaltung einer dienstorientierten Architektur beitragen können. Dafür zeige ich Ihnen im Abschnitt *„Was Sie sich merken sollten“* am Ende des Kapitels eine Beispielanwendung.

In der Frühzeit der Computernutzung bekamen die Benutzer keine direkten Reaktionen von ihren Programmen. Die Entwickler schrieben Code entweder in Maschinensprache oder in etwas sehr Ähnlichem, speicherten diesen auf Medien wie Magnetbändern oder Lochkarten und übergaben ihn dem Computer. Der Computer führte das Programm aus und erstellte eine Ausgabe, entweder weitere Lochkarten oder einen Ausdruck auf Papier. Die Benutzer erhielten das Ausgedruckte als Ergebnis, ohne Verbindung zum Rechner zu haben. Dies wird als „Stapelverarbeitung" bezeichnet, was sich die meisten von uns nicht mehr als Arbeitsmethode vorstellen können.

In modernen Anwendungen nehmen die Benutzer entweder direkt oder über eine Datenzugriffsschicht im System Verbindung mit der Datenbank auf. Sie geben Daten ein und bekommen über einen Bildschirm direkte Rückmeldungen. Dieses Buch wurde mit einem Programm erstellt, das so arbeitet, und möglicherweise lesen Sie es auf dieselbe Weise.

Es gibt jedoch Situationen, in denen die Stapelverarbeitung oder eine verbindungslose Rechnernutzung sinnvoll ist. Bei manchen Anwendungen befindet sich der Benutzer nicht in der Nähe des Computers, wenn die Ergebnisse erstellt werden, benötigt die Daten aber dennoch. Andere Systeme brauchen vielleicht ebenfalls Zugriff auf die Daten, wenn auch nicht sofort. Die Kommunikation zwischen Daten und Benutzern muss nicht direkt erfolgen.

Dieses Paradigma bietet den Vorteil, dass Sie das System über große Gebiete verteilen und die Belastung zwischen den Komponenten ausgleichen können. Wird keine unmittelbare Rückmeldung benötigt, sind geringere Ressourcen erforderlich.

Bei Ihrer täglichen Arbeit benutzen Sie bereits mindestens ein verbindungsloses System. E-Mails werden zum Beispiel zu einer bestimmten Zeit auf einem Computer geschrieben, später auf einem anderen gespeichert und noch später von einem anderen abgerufen und dort gelesen. Der E-Mail-Server stellt Empfang, Speicherung und Auslieferung der Nachricht als Dienst bereit. Im Grunde bilden E-Mails einen Bestandteil einer dienstorientierten Architektur. Darin stellt ein Server Schnittstellen und Daten für jedes Programm bereit, das in der Lage ist, mit dem Dienst zu kommunizieren. Dieser kann auch so konfiguriert werden, dass er Eingaben akzeptiert oder abhängig von der Verbindung eine andere Aktion ausführt.

SOA-Systeme sind nicht neu. Microsoft stellt in Programmen wie BizTalk oder in separaten Add-Ons für SQL Server 2000 zahlreiche SOA-Mechanismen bereit. In SQL Server 2005 wurde das Tool Service Broker direkt in die Engine integriert.

Warum nimmt Microsoft ein SOA-System in SQL Server 2005 auf, wenn solche Systeme bereits in anderen Programmen zur Verfügung gestellt werden? Der Grund liegt in den Voraussetzungen für eine Architektur dieser Art. Wenn Sie verbindungslose Systeme zulassen, muss die dienstorientierte Architektur gewährleisten, dass der Datenverkehr zwischen den Systemen in Nachrichten gekapselt wird, um das sendende Programm zu kennzeichnen, damit es vom Dienst die richtige Antwort erhält. Außerdem müssen Sie dafür sorgen, dass die Nachrichten in der richtigen Reihenfolge angeordnet sind. Das Programm kann zum Beispiel die dritte Zeile einer Bestellung vor dem Header senden. Der Dienst muss die Reihenfolge und die Kapselung kennen, damit er erst dann auf die Bestellung reagiert, wenn sie vollständig ist, genau wie bei einer Datenbanktransaktion.

Um all dies zu verwalten, nutzen SOA-Systeme eine Datenbank. Bei anderen Mechanismen müssen Sie nicht nur das SOA-System, sondern außerdem eine komplette Datenbank verwalten. Indem Sie die dienstorientierte Architektur in das Datenbankmodul integrieren, sind die Daten, die Metadaten, die Daten für die Nachverfolgung und die Mechanismen alle in derselben Architektur angesiedelt. Sie brauchen nur ein einziges System kennenzulernen, zu implementieren und zu verwalten.

Eine weitere Art von dienstorientierter Architektur ist ein „push"-orientiertes Datensystem. Über diese Art der Datenübertragung habe ich schon im letzten Kapitel gesprochen, als es um die Replikationsfunktionen von SQL Server ging. In dieser Umgebung werden die Daten entweder gesendet oder von einer anderen Datenbank abgerufen. Das ist zwar nützlich, aber häufig müssen Sie Daten an einen anderen Empfänger als eine Datenbank senden, beispielsweise an ein E-Mail-Programm oder ein SMS-Gerät. In SQL Server 2005 finden Sie das Tool Notification Services, das genau dafür gedacht ist.

Am Aufbau einer dienstorientierten Architektur sind normalerweise einfach wegen der umfangreichen technischen Fähigkeiten, die für Implementierung und Wartung erforderlich sind, zahlreiche Mitarbeiter Ihrer Technologieabteilung beteiligt. Als Datenbankadministrator werden Sie beauftragt, das System zu verwalten und zu warten und dabei mit den Entwicklern, denjenigen, die die Geschäftsprozesse analysieren, und anderen zusammenzuarbeiten.

7.1 Notification Services

Notification Services gehört zu den Produkten, die tun, was ihr Name sagt: Das Tool stellt ein Instrument bereit, um einen Benutzer über ein Ereignis in einer Datenbank zu informieren. Das System setzt dafür mehrere Komponenten ein, von den Datenbanken, die Daten über die Ereignisse speichern, bis hin zu den Steuerdateien und Datenbanken, die den Vorgang schützen.

Untersuchen wir zunächst einige Anwendungen für Notification Services und ihre Verteilung. Am offensichtlichsten sind die Anwendungen, die unmittelbare Rückmeldungen für einen Benutzer erfordern. Dazu gehören Preis-, Niveau- und Bestandsänderungen sowie weitere zeitabhängige Informationen.

Andere Einsatzgebiete für Notification Services liegen möglicherweise nicht so auf der Hand. Marketingstudien zeigen, dass Kunden sehr frustriert sind, wenn sie das Gefühl haben, dass eine Firma nicht reagiert. Anwendungen lassen sich jedoch so schreiben, dass der Kunde darüber benachrichtigt wird, in welchem Stadium sich seine Bestellung befindet, etwa wann sie zusammengestellt ist, wann sie den Betrieb verlässt oder bei jeder anderen Aktion, die den Statuscode in der Datenbank ändert. Sie können auch eine Benachrichtigung über den Eingang einer Bestellung und das Eingangsdatum senden.

Für den Systemadministrator können Notification Services zur reaktiven Verwaltung des Systems beitragen. Eine reaktive Verwaltung ist im EDV-Bereich nicht ratsam, aber mit Notification Services können Sie eine vorausschauende Logik einsetzen, sodass ein Server bestimmte Objekte beobachtet (beispielsweise Sicherungen oder andere Wartungsarbeiten) und Sie beim Erreichen eines Schwellenwerts benachrichtigt.

Was ist Notification Services nun eigentlich? Notification Services ist eine Infrastruktur von SQL Server 2005, die einen Dienst, Instanzen und Anwendungen sowie eine Reihe von Programmen umfasst, die Ihre Entwickler als Schnittstelle zum System schreiben. In diesem Kapitel stehen die Instanzen und Anwendungen von Notification Services im Mittelpunkt.

Die einfachste Methode zum Anlegen des Systems besteht darin, in den Programmcode Objekte für die Nachrichtenverwaltung (*Notification Management Objects*, NMOs) zu integrieren. Einige Beispiele für diese Art der Programmierung finden Sie, wenn Sie das Beispiel und die Anwendungen auf Ihrem Server installiert haben.

Außerdem können Sie Notification Services mithilfe der nativen Tools von SQL Server 2005 unter Verwendung von XML-Dateien erstellen und verwalten. Wir schauen uns die Struktur dieser Dateien an und untersuchen die Ergebnisse ihrer Implementierung, sodass Sie sehen, wie sie zu verwalten und zu warten sind. In der Praxis sind die meisten Anwendungen mit NMOs geschrieben, Sie können die XML-Dateien für die Instanz und die Anwendung aber in jedem Fall auch über EXPORTIEREN im Kontextmenü des Objekt-Explorers in SQL Server Management Studio anlegen.

7.1.1 Architektur von Notification Services

In der Onlinedokumentation finden Sie eine Menge hervorragender Informationen über Notification Services, die jedoch nicht in Form eines ganzheitlichen Überblicks angeordnet sind. Dafür gibt es einen guten Grund. Die Notification Services nutzen mehrere Komponenten der SQL Server 2005-Plattform für ihre Zwecke. Neben den von Microsoft bereitgestellten Fähigkeiten können Ihre Entwickler jeden Bestandteil des Systems durch benutzerdefinierte Programme und Schnittstellen erweitern. Um ein Bild davon zu gewinnen, wie das alles zusammenpasst, beginnen wir am Ende und arbeiten uns nach vorn zu dem Benutzer durch, der die Benachrichtigung erhält.

Da Notification Services so viele Komponenten hat, unternehme ich drei Durchgänge. Im ersten gebe ich einen groben Überblick, im zweiten erläutere ich das Tool ein wenig detaillierter, und im dritten behandle ich die Grundkomponenten, indem ich die Instanzkonfigurationsdatei (Instance Configuration File, ICF) und die Anwendungungsdefinitionsdatei (Application Definition File, ADF) erkläre. Trotzdem erhalten Sie nur einen groben Überblick über das Thema.

> In Notification Services werden einige Begriffe verwendet, die in SQL Server bereits eine Bedeutung besitzen, wie *Instanz*, *Verleger*, *Abonnent* usw. Achten Sie in diesem Kapitel genau auf die Definitionen, die Notification Services für diese gängigen Begriffe benutzt.

Am rückwärtigen Ende des Vorgangs steht SQL Server 2005, wo die Daten, die die Benutzer sehen müssen, sowie die Metadaten abgelegt werden, die Notification Services für seinen Betrieb benötigt. Oberhalb von SQL Server 2005 befindet sich das Programm `NSServer.exe`, das die Instanzen ausführt, aus denen die Benachrichtigungen für die Benutzer bestehen. Zum Anlegen dieser Instanzen haben Sie zwei Möglichkeiten: Entwickler können sie mithilfe von Programmen erstellen, die NMOs verwenden, Sie können aber auch ein XML-Dokument verwenden, das als Instanzkonfigurationsdatei bezeichnet wird.

In einer Instanz können Sie eine oder mehrere Notification Services-Anwendungen anlegen, die auf Ereignisse lauschen, also auf Änderungen, von denen Ihre Benutzer in Kenntnis gesetzt werden wollen. Das können Datenbankaktivitäten sein, aber auch Änderungen in Analysis Services oder Dateien in einem Dateisystem und andere Vorgänge.

Diese Ereignisse werden mit Abonnements abgeglichen, die den Abonnenten (Benutzern) über verschiedene Protokolle auf Geräte wie Mobiltelefone oder per E-Mail gesendet werden. Wiederum können solche Anwendungen von den Entwicklern mithilfe von NMOs geschrieben oder von Ihnen mithilfe eines XML-Dokuments, einer sogenannten Anwendungsdefinitionsdatei, erstellt werden, die die Bestandteile der Anwendung näher beschreibt.

Dies ist ein erster Überblick, der zahlreiche Details übergeht. Sehen wir uns den Vorgang etwas genauer an. Nachdem die Instanz und die Anwendungen erstellt und in Betrieb genommen worden sind, nutzt die Instanz das Programm `NSServer.exe`, das im Hintergrund ausgeführt wird und mithilfe eines Ereignisanbieters auf Ereignisse lauscht. Ein Ereignisanbieter achtet auf Änderungen an Datenbanken und am Dateisystem sowie auf andere Ereignisse. Dann wendet er Vergleichsregeln an, um die Ereignisse den Abonnements zuzuordnen, und sendet die Daten an einen anderen Anbieter. Dabei handelt es sich um ein Programm, das die Daten entgegennimmt und für die Auslieferung formatiert.

Anbieter werden von der Anwendung aufgerufen, und ein Generator ordnet die Abonnements den Ereignissen zu. Abonnements sind Datengruppen, für die sich ein Abonnent (der Benutzer) interessiert. Notification Services bestückt das gesamte Abonnement, das dann von einem Verteiler gelesen wird. Verteiler setzen besondere Auslieferungsdienste ein, um nicht nur mit anderen Datenbanken, sondern auch mit Dateisystemen, E-Mail-Programmen und Telefonen sowie weiteren Kommunikationssystemen zu kommunizieren, die Ihre Entwickler erstellen.

Der Verteiler formatiert die Ausgabe mit einer Inhaltsformatierung und sendet sie dann an den Anbieter des Auslieferungsprotokolls, der Daten über SMTP, HTTP und SMS übertragen kann. Der Abonnent empfängt die Daten im geeigneten Format über das benötigte Protokoll.

Untersuchen wir als dritte Erläuterung des Vorgangs zwei der wesentlichen Komponenten, mit denen Sie arbeiten werden, noch ausführlicher: die Instanzkonfigurationsdatei und die Anwendungsdefinitionsdatei. Mithilfe dieser XML-Dateien richten Sie die Instanzen und Anwendungen ein, die Ihre Entwickler in die Lage versetzen, ein vollständiges Notification Services-System zu erstellen. Wenn Sie sich diese Dateien ansehen, fügen sich die von mir beschriebenen Teile zu einem Ganzen zusammen.

Instanzen und die Instanzkonfigurationsdatei

Eine Notification Services-Instanz ist eine Sammlung von Anwendungen, die als Einheit auf einem SQL-Server ausgeführt werden. Auf einem Server können mehrere Instanzen betrieben werden. Sie können eine Instanz per Programm mit NMOs anlegen oder mit SQL Server Management Studio und einer Instanzkonfigurationsdatei. Ich verfolge hier den zweiten Ansatz.

Grundlagen der Datenbankadministration: XML

XML ist nicht schwer zu erlernen, wenn Sie es noch nicht beherrschen. Es handelt sich um eine ASCII-Datei mit Sonderzeichen (sogenannten Tags) vor dem Text, den Sie in irgendeiner Weise auszeichnen wollen. Das Prinzip ist dasselbe wie bei HTML, aber die XML-Tags werden nicht von einer unabhängigen Körperschaft festgelegt. Sie können beliebige Tags erstellen. Dabei müssen Sie lediglich daran denken, dass Sie Starttags (im Format `<EinTag>`) und Endtags (im Format `</EinTag>`) verwenden, Headerinformationen hinzufügen und die Tags sorgfältig verschachteln (was dann ein wohlgeformtes Dokument ergibt).

In XML heißen die mit Tags versehenen Wörter *Elemente* und die übrigen Informationen über diese Wörter *Attribute*. Elemente werden von Tags umschlossen, Attribute stehen innerhalb der Klammern. Das folgende Beispiel zeigt das Element `name` mit dem Wert `Buck` und dem Attribut `first`:

```
<name type="first">Buck</name>.
```

Ein weiterer Bestandteil von XML-Dateien ist der Kommentar. Er sieht folgendermaßen aus:

```
<!-- Dies ist ein Kommentar -->
```

Kommentare sind eigentlich eine Art Direktiven. Direktiven sind Sonderzeichen (zum Beispiel Fragezeichen), die den XML-Parser zu bestimmten Handlungsweisen veranlassen (in diesem Fall dazu, den folgenden Text zu ignorieren).

Ein wesentlicher Unterschied zu HTML besteht darin, dass XML zwischen Groß- und Kleinbuchstaben unterscheidet. „Buck“ ist nicht dasselbe wie „buck“. Bei der Verwendung von Tags tappen Sie damit häufig in die Falle.

Über XML lässt sich natürlich noch erheblich mehr sagen, aber schon diese Grundlagen helfen Ihnen beim Lesen der hier vorgestellten Dateien.

Die Instanzkonfigurationsdatei enthält den Namen der Instanz, eine Datenbank, um sie zu steuern, die Namen der Anwendungen, die mit ihr kommunizieren können, sowie Informationen über Verschlüsselung und Auslieferung. Sie können darin auch Versionsinformationen unterbringen, was ich für Produktionsanwendungen empfehle. Das Format der Datei wird in der Onlinedokumentation unter *„Instance Configuration File Reference"* genauer beschrieben.

Hier zeige ich eine Instanzkonfigurationsdatei mit dem Mindestinhalt, damit wir uns über die erforderlichen Bestandteile unterhalten können. In Kürze sehen Sie die Ergebnisse des Imports einer solchen Datei mithilfe von SQL Server Management Studio in die Datenbank. Ein sinnvolles Beispiel für Notification Services finden Sie am ehesten in den Beispielen von Microsoft, wenn Sie sie installiert haben. Ein vollständiges System mit Schnittstelle und Datenbanken einzurichten und zu erklären würde mehr als ein Kapitel erfordern und Sie in Teile des Systems führen, mit denen ein Datenbankadministrator normalerweise nichts zu tun hat. Ich empfehle Ihnen jedoch, die Beispiele zu nutzen, weil sie die Entwicklungszeit verkürzen können und Ihnen die Möglichkeit geben, sich auf die Verwaltung eines Notification Services-Systems zu konzentrieren.

Der folgende Code stellt die Minimalversion dar, die Sie für eine einfache Instanz bereitstellen müssen. Die einzelnen Abschnitte sind kommentiert (`<!-- Kommentar -->`), damit wir sie leichter erörtern können:

```
<?xml version="1.0" encoding="utf-8"?>
<NotificationServicesInstance
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.microsoft.com/MicrosoftNotificationServices
/ConfigurationFileSchema">
<!-- Notification Services instance Name -->
<InstanceName></InstanceName>
<!-- Database Engine instance -->
<SqlServerSystem></SqlServerSystem>
<!-- Applications -->
<Applications>
<Application>
<ApplicationName></ApplicationName>
<BaseDirectoryPath></BaseDirectoryPath>
```

```
<ApplicationDefinitionFilePath></ApplicationDefinitionFilePath
>
</Application>
</Applications>
<!-- Delivery Channels -->
<DeliveryChannels>
<DeliveryChannel>
<DeliveryChannelName></DeliveryChannelName>
<ProtocolName></ProtocolName>
</DeliveryChannel>
</DeliveryChannels>
</NotificationServicesInstance>
```

Es sieht aus, als wäre hier eine Menge zu tun, aber eigentlich ist es ganz einfach. Die ersten fünf Zeilen der Datei bilden einen XML-Header, der beschreibt, was die Datei ist und wie sie verwendet werden soll. Dies ist für alle Ihre Instanzen gleich. Die übrigen Abschnitte steuern ähnlich wie die alten `INI`-Dateien früherer Windows-Programme das Anlegen der Instanz.

<!-- Notification Services instanceName -->

Der erste, im Element `<instanceName>` beschriebene Abschnitt legt den Instanznamen für das System sowie den Dienst fest, der zur Ausführung dieser Instanz angelegt wird. Von hier an verweisen die Anwendungen und übrigen Teile dieser Gruppe des Benachrichtigungssystems auf den Namen. Normalerweise halte ich ihn so kurz und aussagekräftig wie möglich. Der Instanzname wird mit dem Präfix `NS$` zum Namen des Dienstes versehen, der im Betriebssystem ausgeführt wird. Der folgende Eintrag legt zum Beispiel den Dienst `NS$AWInventory` an:

```
<InstanceName>AWInventory</InstanceName>
```

<!-- Database Engine instance -->

Das Element `<SqlServerSystem>` enthält den Namen der SQL Server-Instanz (dies hat nichts mit der Notification Services-Instanz zu tun, die mit dieser Datei eingerichtet wird), unter der die Notification Services-Instanz ausgeführt wird. Es ist der Name des Servers, auf dem der Dienst ausgeführt werden soll.

<!-- Applications -->

Jede Instanz enthält eine oder mehrere Anwendungen, die mit ihr kommunizieren können. Das Element `<Applications>` ist ein übergeordnetes Element mit weiteren Informationen über die Anwendungen, die unter dieser Instanz ausgeführt werden. Es zeigt auf die Anwendungsdefinitionsdatei, die Sie vor dem Importieren der Instanz mithilfe der Instanzkonfigurationsdatei noch anlegen müssen.

> Wenn Sie versuchen, eine Instanzkonfigurationsdatei zu importieren, bevor die Anwendungsdefinitionsdatei vollständig ist, wird ein Fehler gemeldet.

Dieser Abschnitt enthält den Namen der Anwendung sowie ihren Namen und Speicherort auf der Festplatte. Sie können die Dateien in einem gemeinsam genutzten Ordner ablegen, solange die SQL Server- und die Dienstkonten darauf Zugriff haben.

> **Grundlagen der Datenbankadministration: XML-Hierarchien**
>
> Wenn Sie in der Datei weitergehen, bis Sie das öffnende Tag `<Applications>` und das schließende Tag `</Applications>` sehen, stellen Sie fest, dass dazwischen weitere Elemente stehen. Dies entspricht der Verschachtelung von Elementen in XML, um sie anderen Elementen unterzuordnen.

In diesem Beispiel lege ich nur eine Anwendung für die Instanz an, aber es sind so viele möglich, wie die Systemressourcen hergeben. Am besten planen Sie das gesamte System, bevor Sie mit diesem Vorgang beginnen, damit die Instanzen logisch angeordnet werden und jeweils die Anwendungen enthalten, die sinnvoll in einer Gruppe zusammengefasst werden.

Das erste untergeordnete Element im Tag `<Applications>` heißt `<Application>` und ist selbst gleichzeitig ein übergeordnetes Element. Sie wiederholen es für jede Anwendung, die die Instanz beherbergt. Innerhalb dieses Elements steht das Element `<ApplicationName>`, das den Namen der betreffenden Anwendung angibt.

Jede Anwendung benötigt ein Verzeichnis, um ihre Anwendungsdefinitionsdatei zu speichern. Diese Datei, die ich in Kürze anlege, enthält dieselbe Art von Informationen wie die für die Instanz, betrifft aber die Anwendung.

Das nächste Element, `<ApplicationDefinitionFilePath>`, nennt den Namen der Anwendungsdefinitionsdatei. Wenn Sie hier nur den Dateinamen angeben, muss sie direkt am Speicherort `BaseDirectoryPath` gespeichert werden.

Der Grund, warum zwei Elemente verfügbar sind, besteht darin, dass Sie in der Produktion normalerweise mehrere Anwendungen in einer einzigen Instanz ausführen. Die Anwendungen sollen aber voneinander getrennt bleiben, damit den Entwicklern nur auf diejenigen Anwendungen Zugriff ermöglicht werden kann, mit denen sie tatsächlich arbeiten. Die Unterbringung der Anwendungen in getrennten Verzeichnissen ermöglicht eine höhere Sicherheitsstufe.

<!-- Delivery Channels -->

Die Instanz steuert die Art der Kommunikation des Systems mit der Außenwelt. Das erste übergeordnete Element am Anfang des Abschnitts, `<DeliveryChannels>`, beschreibt alle Kommunikationsmethoden für diese Instanz.

Sie öffnen die Übertragungsmethode mit dem Element `<DeliveryChannel>` und geben dann das Element `<DeliveryChannelName>` ein, um den gewünschten Typ festzulegen, von E-Mail bis Datei. Innerhalb dieser Anweisung müssen Sie festlegen, welches Protokoll der Kanal benutzen soll. Dazu dient der Wert des Elements `<ProtocolName>`.

Es gibt zahlreiche Möglichkeiten, Informationen mit jeweils eigenen Protokolleinstellungen zu senden. Damit wir den Überblick hier fortsetzen können, verweise ich Sie auf die bereits erwähnte Themensuche der Onlinedokumentation, wenn Sie mehr über die verfügbaren Methoden erfahren wollen.

Nachdem die Grundelemente definiert sind, müssen wir die Anwendungsdefinitionsdatei erstellen.

Anwendungen und die Anwendungsdefinitionsdatei

In einer einzigen Notification Services-Instanz sind eine oder mehrere Anwendungen enthalten. Wenn Sie eine Anwendung erstellen, legen Sie auch eine Datenbank an (oder geben sie an), in der die Ereignisse, die Abonnements, die Benachrichtigungsdaten und weitere Metadaten über die Anwendung abgelegt werden. Ist keine Datenbank vorhanden, können Sie eine über die XML-Datei (oder das NMO-Programm) anlegen lassen. Werfen wir wieder einen Blick auf eine einfache XML-Datei, und untersuchen wir die darin enthaltenen Elemente:

```
<?xml version="1.0" encoding="utf-8" ?>
<Application xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.microsoft.com/MicrosoftNotificationServices
/ApplicationDefinitionFileSchema">
<!-- Version -->
```

```
<!-- History -->
<!-- Database Definition -->
<!-- Event Classes -->
<!-- Subscription Classes -->
<SubscriptionClasses></SubscriptionClasses>
<!-- Notification Classes -->
<NotificationClasses></NotificationClasses>
<!-- Event Providers -->
<!-- Generator Settings -->
<Generator>
<SystemName>%SystemName%</SystemName>
</Generator>
<!-- Distributor Settings -->
<Distributors>
<Distributor>
<SystemName>%SystemName%</SystemName>
</Distributor>
</Distributors>
<!-- Application Execution Settings -->
<!-- Wichtig: Sie sollten mindestens einen Leerungszeitplan
definieren und die Verteilerprotokollierung ganz oder teilweise
ausschalten. -->
</Application>
```

Diese Datei weist nur einige wenige Elemente auf, weshalb ich Ihnen auch hier empfehle, sich in der Onlinedokumentation unter „Application Definition File Reference“ darüber zu informieren. Dort finden Sie sämtliche Elemente mit Hyperlinks versehen in alphabetischer Reihenfolge. Mit der hier abgedruckten Minimalvorlage und Ihren Suchergebnissen können Sie die gewünschte Datei schnell zusammenstellen.

In dieser knappen Datei legen die ersten vier Zeilen den Header der XML-Datei fest und teilen SQL Server 2005 mit, wie sie gehandhabt werden soll.

<!-- Version -->

Der erste Abschnitt umfasst das Element <Version>, das mehrere untergeordnete Elemente mit Versions- und Buildnummern enthält. Es ist optional, aber ich verwende es normalerweise, um verfolgen zu können, an welcher Stelle im Buildvorgang ich mich befinde. Das ist sicher eine gute Angewohnheit.

<!-- History -->

Dieser Abschnitt ähnelt dem vorhergehenden insofern, als er Nachverfolgungsdaten für Ihr System bereitstellt. Sie können hier Daten aufnehmen, um Änderungen an der Anwendungsdefinitionsdatei zu verfolgen.

<!-- Database Definition -->

Mit den Elementen des Abschnitts <DatabaseDefinition> legen Sie den Namen, das System und die physischen Strukturen für die Anwendung fest. Verzichten Sie darauf, erstellt das System eine Datenbank aus den Standardwerten des Servers. Demnächst lege ich eine Notification Services-Anwendung an und zeige Ihnen, was in dieser Datenbank abgelegt wird.

<!-- Event Classes -->

In diesem Abschnitt wird es wirklich interessant. Ausgehend vom übergeordneten Element <EventClasses> richten Sie ein oder mehrere <EventClass>-Elemente ein, die definieren, auf welche Elemente das System reagiert.

<!-- Subscription Classes -->

In diesem Abschnitt gehen Sie vom übergeordneten Tag <SubscriptionClasses> aus und richten mithilfe von <SubsriptionClass>-Elementen ein oder mehrere Abonnements für die Anwendung ein. Hier geben Sie den Namen, das Abonnementssystem, die gewünschten Indizes für die Tabellen sowie Ereignis- und Zeitplanregeln an. Damit wird festgelegt, wie oft die Ereignisse aufgezeichnet werden und was abonniert werden kann.

<!-- Notification Classes -->

Unter dem übergeordneten Tag <NotificationClasses> richten Sie mithilfe von <NotificationClass>-Elementen eine oder mehrere Benachrichtigungen für die Anwendung ein. Die Elemente definieren, wo die Anwendung die Benachrichtigungen ablegt, welcher Filter für ihre Formatierung verwendet wird und welche Protokolle zum Einsatz kommen.

Außerdem können Sie festlegen, ob die Benachrichtigungen einzeln gesendet werden oder eine zusammenfassende Übersicht erstellt und erst nach Ablauf einer bestimmten Zeit übertragen wird.

Diese Einstellungen ähneln denen, die Sie in einer Newsgroup finden. Wenn Sie sich die Benachrichtigungen so vorstellen, haben Sie das Prinzip erfasst.

<!-- Event Providers -->

Wie bereits erläutert, ist ein Ereignis eine Änderung der für Sie interessanten Daten, beispielsweise die Änderung eines Aktienkurses oder eines Projekts. Microsoft stellt in SQL Server 2005 einige Ereignisanbieter zur Verfügung, Sie können jedoch auch eigene schreiben. Die folgende Liste enthält einige Ereignisanbieter, die Ihr System einsetzen kann.

Tabelle 7.1: Ereignisanbieter in SQL Server 2005

Analysis Services	MDX-Anweisungen
File Watcher	XML-Ausgaben; setzt ein Schemadokument voraus.
SQL Server	Führt eine SQL-Anweisung aus, um Veränderungen zu erkennen; kann auch von einem anderen Anbieter als Microsoft sein, weil verbundene Server verwendet werden.
Benutzerdefiniert	Nutzt `IEventProvider` und `IScheduleEventProvider`.

Die vorherrschende Rolle spielt nach meiner Erfahrung das SQL Server-Ereignis. Es ermöglicht Ihnen, eine Tabelle auf Änderungen zu überwachen und die Informationen mittels Abonnements an einen Benutzer zu übermitteln.

Jeder Ereignistyp setzt bestimmte Elemente voraus, um verarbeitet werden zu können.

<!-- Generator Settings -->

Der Generator ist die Notification Services-Komponente, die Datenänderungen (Ereignisse) denen zuordnet, die davon erfahren sollen (den Abonnenten). Er wird auf Ihrem Server mithilfe des Programms `NSServer.exe` ausgeführt.

In diesem Abschnitt definieren Sie mithilfe von Elementen den Namen des Systems, das als Generator fungiert, und legen fest, wie viele Threads er benutzt.

<!-- Distributor Settings -->

Der Verteiler ist die Notification Services-Komponente, die die Daten formatiert und sendet. Sie können ihn auf einem oder mehreren Servern speichern, um die Systemauslastung zu verteilen. Mithilfe von Elementen legen Sie den Namen des Verteilers, die verwendeten Threads und eine optionale Einstellung für die Dauer fest.

<!-- Application Execution Settings -->

In diesem Abschnitt geht es um die Informationen, die die Verarbeitung von Ereignissen betreffen, beispielsweise um die Beschränkung der Zeit, die für einen bestimmten Task aufgewendet wird, und um die Reihenfolge, in der die Prozesse verarbeitet werden. Ein wichtiger Wert ist `<Vacuum>`, der besagt, wie oft eine Bereinigung stattfindet.

Sehen wir uns vor diesem Hintergrund an, wie dies alles im Rahmen einer ganzen Anwendung zusammenwirkt. Ich habe vor, eine einzige Anwendung zur Beobachtung von Bestandsdaten einzurichten. Dazu habe ich die Instanzkonfigurationsdatei und die Anwendungsdefinitionsdatei erstellt und in einem Verzeichnis auf meinem lokalen Testsystem abgelegt. Ich will sie in meine SQL Server 2005-Instanz importieren.

Installieren der Instanz und Erstellen der Anwendung

In Abbildung 7.1 habe ich SQL Server Management Studio gestartet und mit der rechten Maustaste auf NOTIFICATION SERVICES im Objekt-Explorer geklickt und NEUE NOTIFICATION SERVICES-INSTANZ ausgewählt.

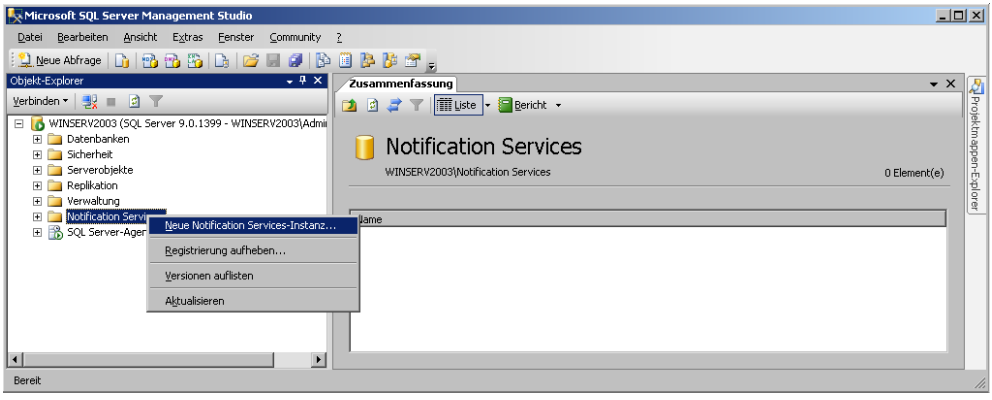

Abbildung 7.1: Instanz des Notification-Service erstellen

In diesem Fenster lege ich den Speicherort der Instanzkonfigurationsdatei an, wie Sie in Abbildung 7.2 sehen.

Danach klicke ich auf OK, um die Datei zu verarbeiten. Mein System zeigt während dieses Vorgangs das Fenster in Abbildung 7.3 an.

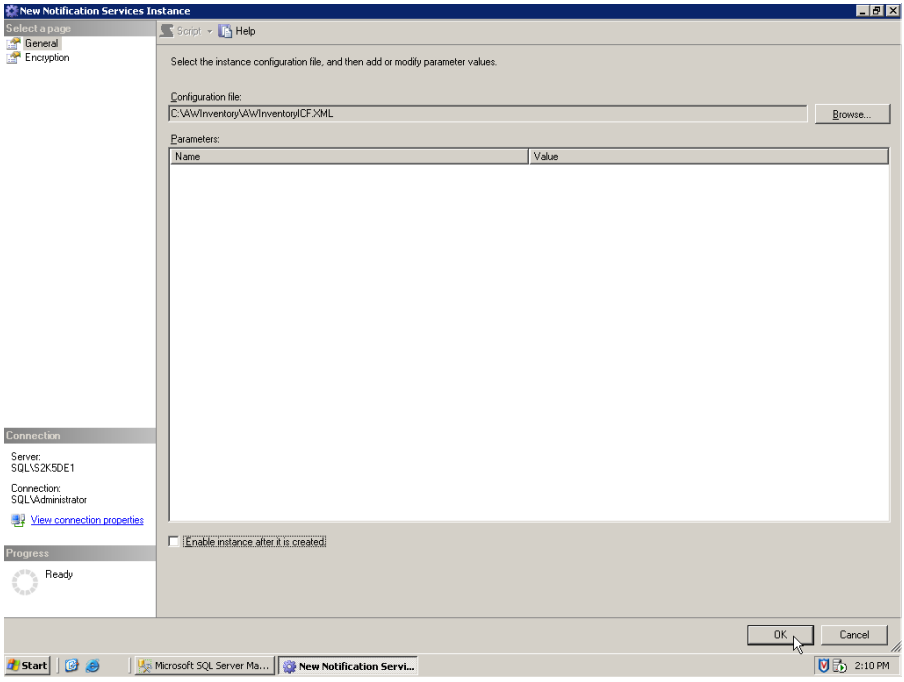

Abbildung 7.2: Speicherort der Instanzkonfigurationsdatei festlegen

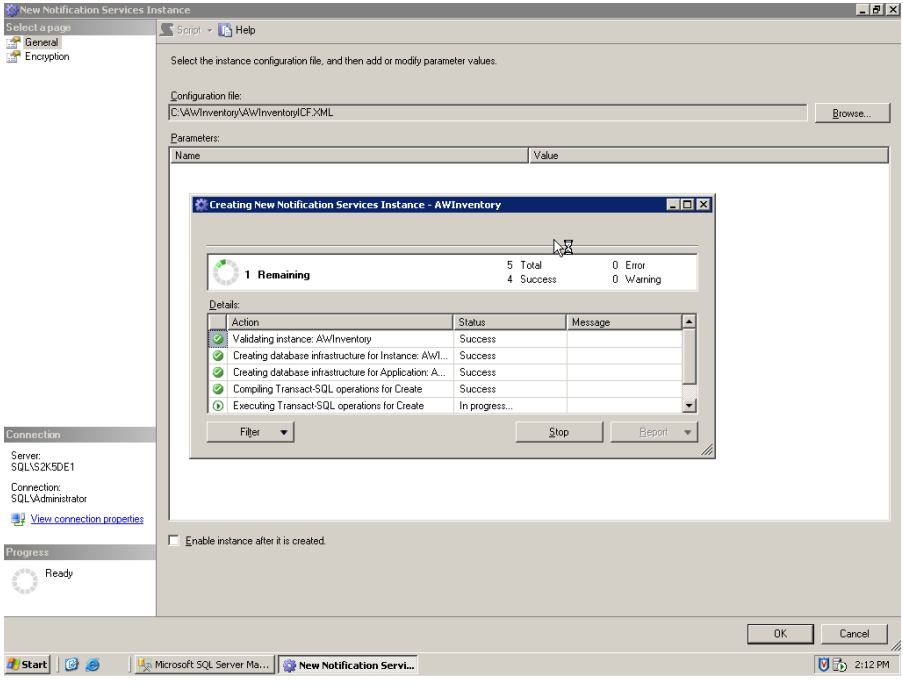

Abbildung 7.3: Status der Instanzerstellung anzeigen

Jetzt aktiviere ich das System, indem ich mit der rechten Maustaste im Objekt-Explorer noch einmal darauf klicke. Damit werden die Verteiler, die Generatoren und die Abonnements für das System festgelegt. Das Ergebnis sehen Sie in Abbildung 7.4.

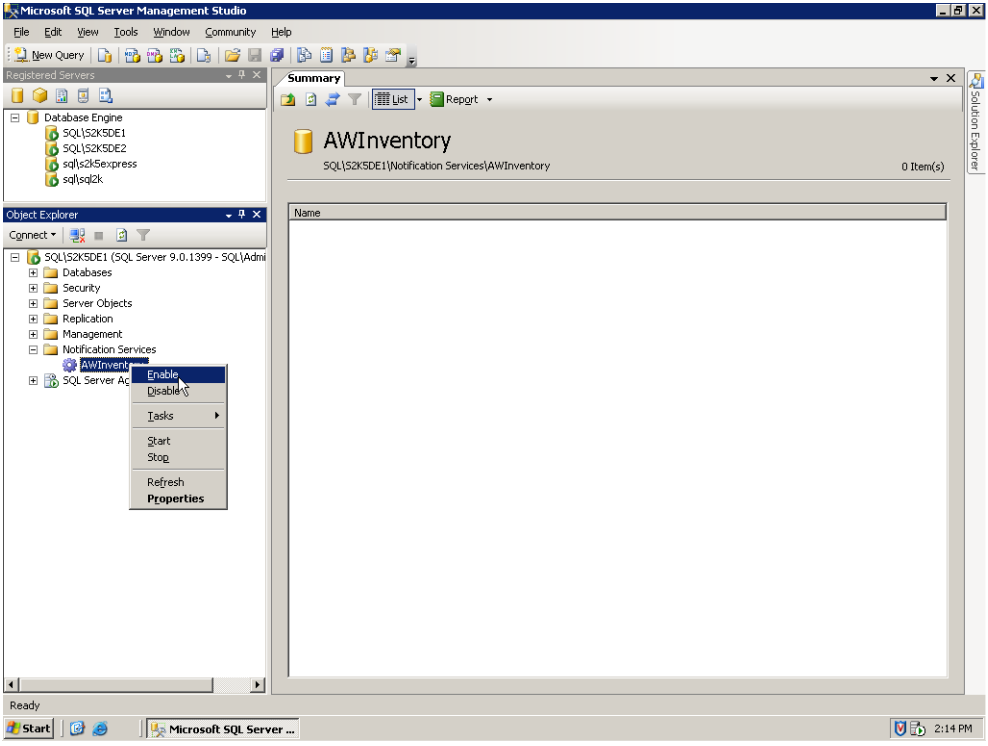

Abbildung 7.4: Instanz aktivieren

Danach klicke ich mit der rechten Maustaste auf den Dienst und wähle im Kontextmenü erst TASKS und dann REGISTRIEREN (siehe Abbildung 7.5).

Damit gelange ich in ein Fenster, in dem ich den Dienst erstelle sowie die Autorisierung für ihn und die Art seines Zugriffs auf die Datenbank festlege (siehe Abbildung 7.6).

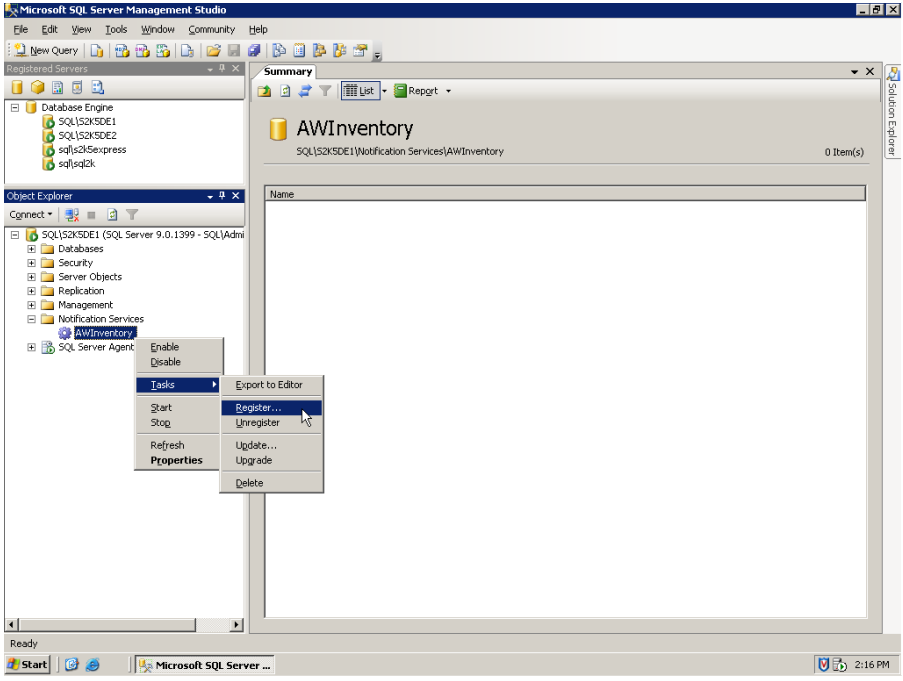

Abbildung 7.5: Tasks registrieren

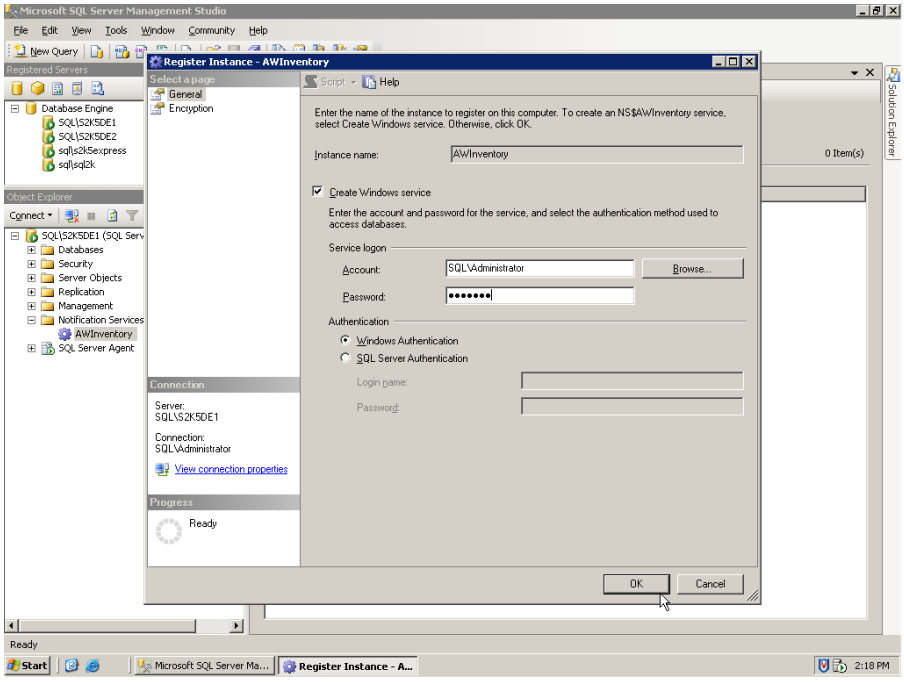

Abbildung 7.6: Dienst für Instanz erstellen und konfigurieren

Ich erhalte Rückmeldungen, die zeigen, dass das System Registrierungseinträge anlegt, den Dienst erstellt und neue Leistungsindikatoren hinzufügt. Wenn alles abgeschlossen ist, klicke ich wieder mit der rechten Maustaste auf den Namen des Dienstes und wähle im Kontextmenü STARTEN. Nachdem ich den Start bestätigt habe, werden die in der Instanzkonfigurationsdatei und der Anwendungsdefinitionsdatei angegebenen Datenbanken (`AWInventoryNSMain` bzw. `NSMetadata`) angelegt und wird der Dienst gestartet.

Abbildung 7.7 zeigt eine Liste der Tabellen in der Instanzdatenbank `AWInventoryNSMain`.

Diese Datenbank enthält die Elemente des Instanzkonfigurationsdatei-Dokuments, die ich bereits erläutert habe. Außerdem gibt es hier weitere Tabellen mit Metadaten, die insbesondere die Zeiteinstellungen wie Zeitzone und Gebietsschema betreffen.

Die Tabellen in der Anwendungsdatenbank `NSMetaData` enthalten ebenfalls die Elemente, die ich bereits vorgestellt habe, sowie weitere Metadaten. Sie sehen dies alles in Abbildung 7.8.

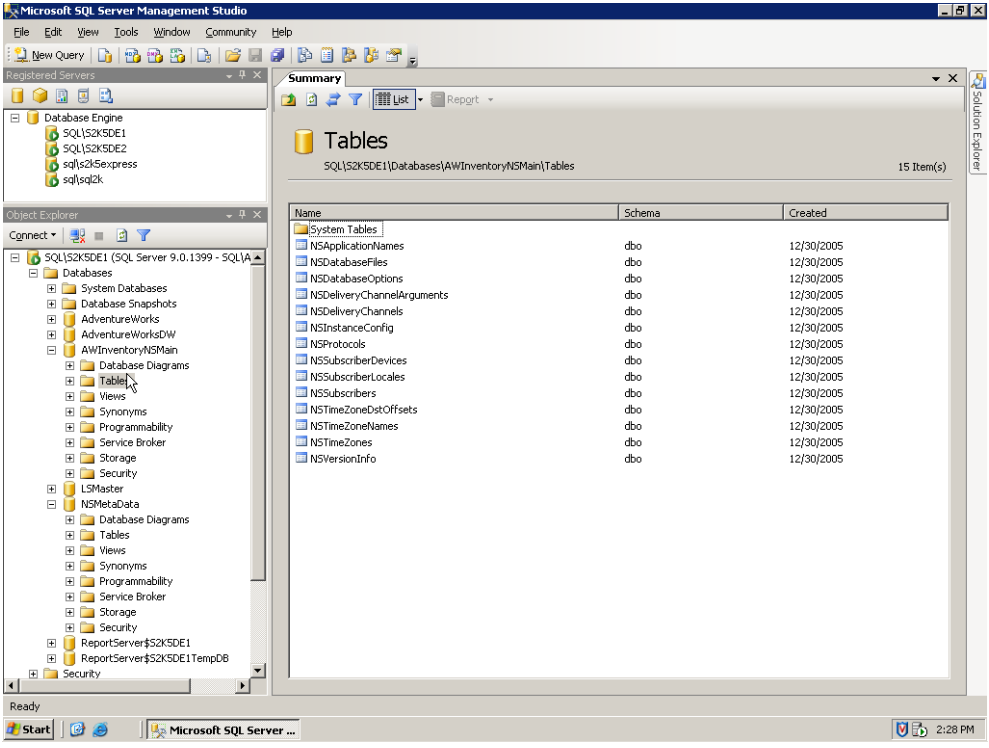

Abbildung 7.7: Tabellen einer Instanzdatenbank

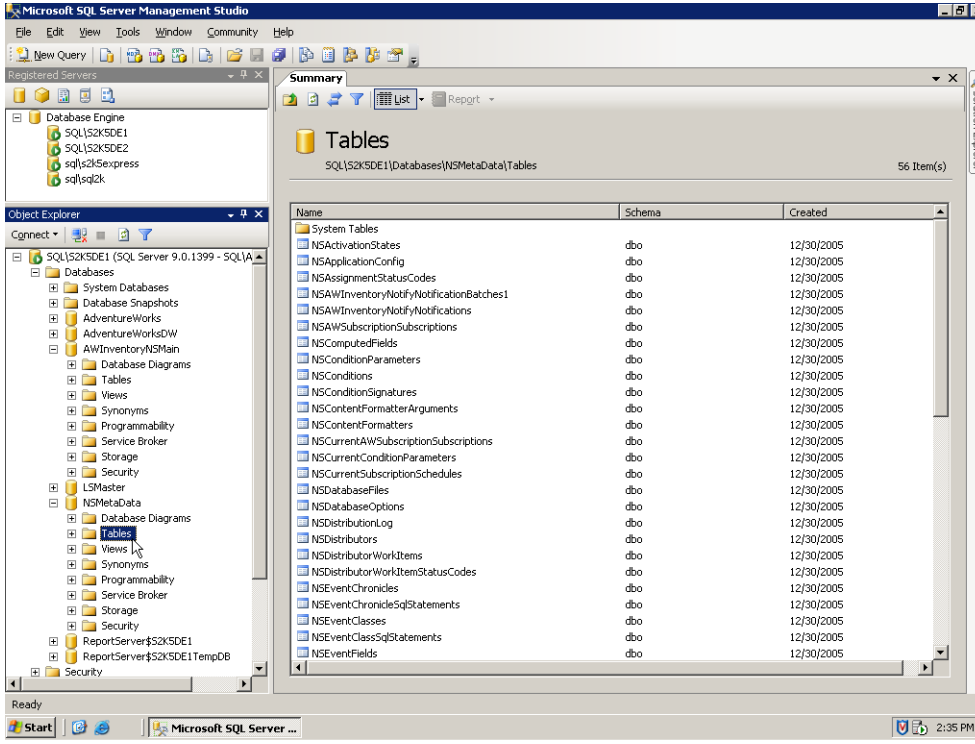

Abbildung 7.8: Tabellen der Anwendungsdatenbank

Der Dienst liest diese Datenbanken ein und greift auf die Daten zu. Damit ist er für die Abonnements der Clientprogramme bereit, ähnlich wie ich es für die Replikation erläutert habe.

7.1.2 Sicherheit

Die Sicherheit Ihres Notification Services-Systems besteht aus zwei Teilen. Der erste Teil umfasst den Schutz der Stellen, an denen sich die Instanzkonfigurations- und die Anwendungsdefinitionsdatei befinden.

Diese Dateien müssen geschützt werden, weil diejenigen, die Zugriff darauf haben, sie so ändern können, dass im Abschnitt `Events` mehr Ereignisse angezeigt werden, als Sie vielleicht wünschen. Die knappen Dateien, die ich beschrieben habe, sind nicht repräsentativ für diejenigen in einer Produktionsumgebung. Diese sind wesentlich größer, sodass Sie ein neues Ereignis oder die Änderung eines aktuellen Ereignisses übersehen können.

Wenn Ihre Instanz läuft, können Sie sie ändern, indem Sie die Instanzkonfigurations- und die Anwendungsdefinitionsdatei erneut importieren. Dies ist die Stelle, an der Gefahr droht, wenn Sie die Bereiche

ungeschützt lassen. Wenn Ihre Entwickler zum Programmieren der Anwendung nicht die Steuerdateien verwenden, sondern NMOs, ist das Problem geringer.

Der zweite Teil des Systemschutzes betrifft die Abonnementdaten. Diese schützen Sie einfach mithilfe von Windows- oder SQL Server-Konten.

7.1.3 Überwachung und Leistungsoptimierung

Notification Services kann auf skalierbaren Servern und auch in einem Cluster eingesetzt werden. Eine weitere Optimierungsmethode für die Anwendung besteht in der Verteilung auf mehrere Server, einschließlich der Generatoren und anderer Teile. Deshalb ist es wichtig, das System im Voraus zu planen.

Sie können ein Notification Services-System in SQL Server Management Studio überwachen, indem Sie mit der rechten Maustaste auf die gewünschte Instanz klicken und EIGENSCHAFTEN wählen. Dann klicken Sie im linken Fenster auf ANWENDUNGEN und markieren unter KOMPONENTEN die Einträge der Spalte AKTIVIEREN (siehe Abbildung 7.9).

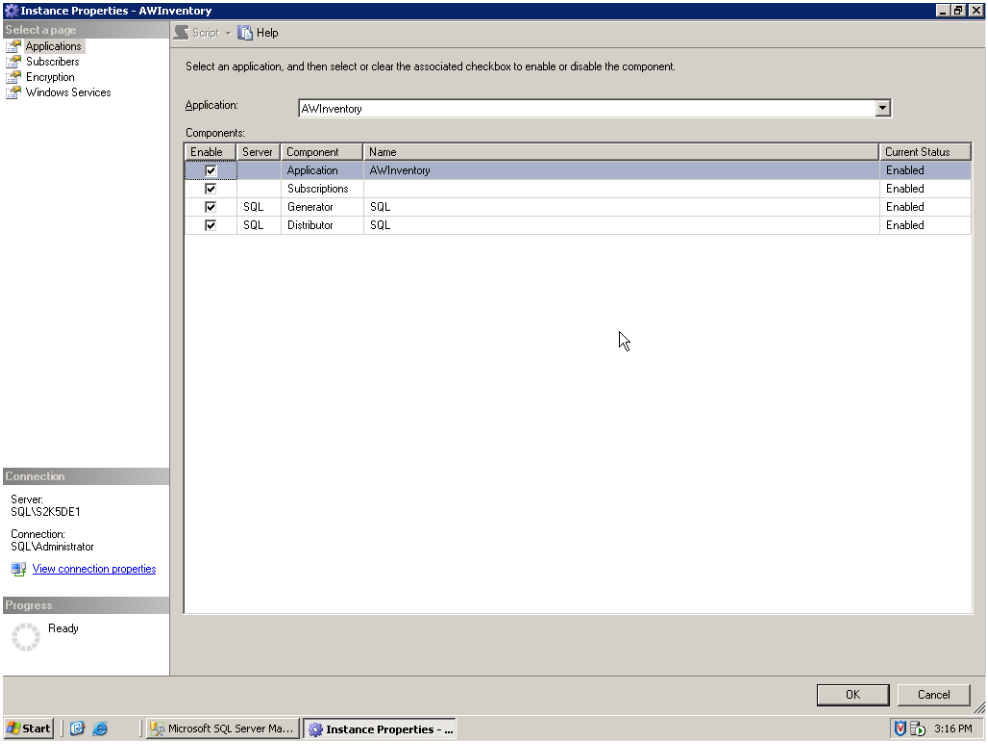

Abbildung 7.9: Notification-Services überwachen

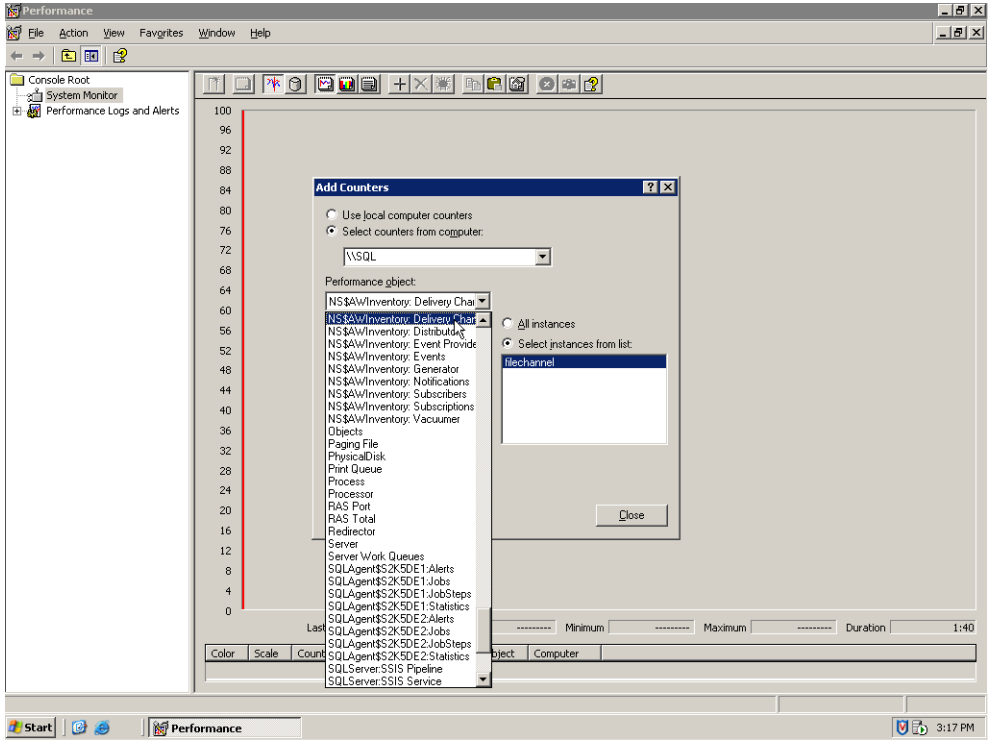

Abbildung 7.10: Leistungsindikatoren für Notification-Services hinzufügen

Außerdem können Sie in diesem Ressourcendialogfeld den Status des ABONNENTEN überprüfen. Ein weiteres Überwachungstool bilden verschiedene Leistungsobjekte und -indikatoren, auf die Sie vom Windows-Systemmonitor aus zugreifen können (siehe Abbildung 7.10).

Alle Teile des Systems lassen sich nach Wunsch beobachten, vom Abonnentensubsystem bis hin zu den Erfassungsmethoden.

7.2 Service Broker

Bei der Replikation werden Daten von einem Server auf einen anderen exportiert, verbleiben aber immer im Datenbankformat. In Notification Services werden Daten per Telefon, E-Mail oder in anderen Formaten an Abonnenten übertragen. Zwar lassen sich Daten mithilfe dieser Technologien auch zwischen Systemen hin- und hersenden, aber häufig werden sie nur in eine Richtung eingesetzt, indem Daten von einem Server an ein anderes System oder einen Benutzer übertragen werden. Ich habe bereits erwähnt, dass die Verwendung eines verbindungslosen Systems wie Replikation und Notification Services, die im Grunde Systeme zum Speichern und Weiterleiten

darstellen, gewisse Vorteile bietet. Der Service Broker von SQL Server 2005 erweitert dieses Konzept auf die gesamte Anwendung.

Mit Service Broker können Sie eine ganze Infrastruktur aufbauen, die „lose gekoppelt" ist, d.h., dass das System keine unmittelbare Rückmeldung von den einzelnen Anwendungen über den Abschluss der Tasks bekommt, sondern nur über deren Eingang.

Microsoft verwendet zum Erklären von Service Broker die Post als Analogie. Dass ist eine gute Veranschaulichung, um dem Datenbankadministrator eine dienstorientierte Architektur vorzustellen. Ich beschreibe mithilfe dieser Analogie die Vorgänge in einem Postamt und übertrage sie dann auf die Bestandteile von Service Broker. Im Abschnitt *„Was Sie sich merken sollten"* zeige ich Ihnen ein einfaches Beispiel für ein Service Broker-Programm, das Sie erweitern können, damit es auf Ihren Produktionsservern eingesetzt werden kann.

7.2.1 Die Post als Veranschaulichung für Service Broker

Nehmen wir an, Sie wollen per Post eine Bestellung für einen neuen Rechner aufgeben. Sie schreiben einen Brief, stecken ihn in einen Umschlag mit Anweisungen für den Bestimmungsort (die Adresse) und werfen ihn in den Briefkasten. Der Briefkasten wird geleert und der Inhalt in das lokale Briefzentrum gebracht, sortiert und zum nächsten Briefzentrum transportiert, bis die Sendung schließlich die für den Empfänger zuständige Verteilstelle erreicht. Diese stellt dem Empfänger die Nachricht zu. Er öffnet den Brief und liest Ihre Anweisungen. Sie lösen mehrere Aktionen in verschiedenen Abteilungen aus, erhalten eine Bestätigung und später ein Paket von der Firma, das die Transaktion abschließt.

Die meisten von uns nutzen täglich solche Systeme, ohne über ihre Vorteile nachzudenken. Einerseits bieten sie eine Verteilung von Funktionen. Als Besteller schreiben Sie die Nachricht und empfangen die Ware, ohne zu wissen oder sich darum zu kümmern, welchen Weg Ihr Schreiben nimmt. Die Postlogistik übernimmt den Umschlag, ohne seinen Inhalt oder sein Ziel zu kennen. Das Briefzentrum sortiert die Nachricht und leitet sie weiter, ohne ihren Inhalt zu kennen. Der Empfänger liest sie, ohne sich für ihren bisherigen Weg oder ihr späteres Ziel zu interessieren. Anschließend wird der Vorgang umgekehrt. Solange jeder seine Rolle spielt, gibt es keinen Bedarf für ein aktives Lenkungssystem, das jede Komponente vom Weg bis zum Inhalt der Nachricht kennen muss. Mit anderen Worten: Niemand braucht den Brief von Anfang bis Ende zu verfolgen.

Ein weiterer Vorteil besteht darin, dass eine einzelne Nachricht beim Empfänger möglicherweise eine Menge an Aktivitäten auslöst. Wenn Sie etwas per Post bestellen, setzen Sie mehrere Abteilungen der

Firma in Bewegung, beispielsweise Einkauf, Auslieferung, Marketing usw. Alles wegen einer einzigen kleinen Bestellung! Dabei brauchen Sie nicht an jede einzelne Abteilung zu schreiben, damit sie Ihre Bestellung bearbeitet, sondern jede Abteilung bearbeitet mehrere Bestellungen gleichzeitig.

Genauso funktioniert Service Broker. Jede Komponente hat eine bestimmte Funktion, wobei Elemente an jedem Punkt mit der Garantie eingeschleust werden können, dass die Komponente ihre Funktion erfüllt, ohne einen der anderen Teile zu kennen oder zu steuern. Die Entwickler können komplexe Systeme schreiben, indem sie die einzelnen Bestandteile für Service Broker programmieren und jeweils gleichzeitig an mehreren Dingen arbeiten lassen.

Untersuchen wir die einzelnen Service Broker-Teile in Hinblick auf die Postanalogie. Dabei erläutere ich die Ähnlichkeit zwischen Service Broker und der Post und gebe Ihnen Informationen, die Sie zum Implementieren und Warten Ihres Systems benötigen.

Genau wie Notification Services führt auch Service Broker eine vollkommen andere Verwendung derselben Terminologie als bei der Replikation und sonst in der EDV ein. Achten Sie in diesem Abschnitt auf neue Definitionen für Begriffe wie *Typ* und *Dienst*.

Service Broker und die Post

Als Sinnbild für den gesamten Vorgang dient ein Postsystem. Jemand stellt zu Beginn die Regeln für die Beförderung von Nachrichten vom Absender zum Empfänger auf (in den USA war das Benjamin Franklin). Was Service Broker betrifft, arbeitet das SQL Server 2005-Datenbankmodul in Form von Konversationen, die einer unidirektionalen Bewegung im Postsystem ähneln. Die Postlogistik nimmt zum Beispiel Ihren Brief aus dem Kasten und transportiert ihn zum Briefzentrum. In der Service Broker-Terminologie wird dies als Konversation bezeichnet.

Glücklicherweise entspricht die Nachricht im Service Broker der Nachricht im Postsystem, wobei beides sowohl die Umhüllung (den Umschlag) als auch den enthaltenen Text bezeichnet. Die einzige Schwierigkeit besteht darin, dass Sie unbedingt definieren müssen, ob Sie über das Ganze oder nur über den Inhalt sprechen. Ich versuche, dies im weiteren Verlauf zu verdeutlichen.

In einem einfachen Beispiel, bei dem sich der Empfänger in derselben Stadt wie Sie befindet, verlässt die Nachricht das Briefzentrum und wandert in den Briefkasten des Empfängers. Das ist eine weitere Konversation. Zusammen bilden die beiden Gespräche und die Nachricht eine sogenannte Konversationsgruppe.

In T-SQL können Sie die Konversationsgruppe mit dem Befehl `GET CONVERSATION GROUP` einsehen. Häufig verwenden Sie diesen Befehl, um den Status eines bestimmten Systemteils abzufragen.

Wenn Ihr Brief beim Briefzentrum ankommt, wird er bis zur weiteren Beförderung dort abgelegt. In Service Broker wird dies von einem Speichermedium namens Warteschlange erledigt.

Einmal im Briefzentrum, bewegt sich der Brief von einer Stelle zur anderen, um richtig sortiert zu werden. Diese internen Vorgänge, die prüfen, ob sich alles an der richtigen Stelle befindet, heißen in Service Broker *Dialoge*. Das Briefzentrum stempelt den Brief ab, um zu dokumentieren, wann er eingegangen ist. Bei Behördenbriefen, Preisausschreiben usw. ist oft das Datum des Poststempels entscheidend. Service Broker-Dialoge tun dasselbe. Sie versehen jede Nachricht mit einer fortlaufenden Nummer, damit ihre Korrektheit gewährleistet ist. Dies ist für transaktionsorientierte Anwendungen wichtig, weil eine Zeile die vorhergehende ändern kann.

Im Briefzentrum sind für das Weiterleiten, Sortieren und Ausliefern mehrere Personen angestellt. Die Mitarbeiter von Service Broker heißen Dienste und werden durch Aktivierungen zum Arbeiten veranlasst. Dies besprechen wir in Kürze eingehender.

Eigentlich verarbeitet das Briefzentrum mehrere Arten von Sendungen. Sie können eine Postkarte oder Briefe unterschiedlicher Größe verschicken. Es kann auch ein Päckchen sein, und Sie können sogar Vorausverfügungen angeben. In Service Broker wird dies im Nachrichtentyp zusammengefasst. Alles, was die Anwendungen senden können, von Text bis zu Musikdateien, wird je nach Typ unterschiedlich behandelt. Die meisten Probleme, die ich mit Service Broker erlebt habe, waren auf eine falsche Typisierung zurückzuführen.

Schließlich wird der Brief vom Empfänger gelesen. In Service Broker sendet das System eine Dialogendenachricht, die die Konversation abschließt.

Die Post: Nachrichtentypen und Inhalte

Der Postvorgang beginnt mit der Postnachricht selbst. Eine Postsendung besteht aus mindestens zwei Teilen: dem Umschlag und der darin enthaltenen Nachricht. Der Umschlag hat die geforderten Maße, muss an bestimmten Stellen die Absender- und die Empfängeradresse tragen, lesbar und ausreichend frankiert sein. Sie schließen einen Vertrag mit der Post ab, der besagt, dass Sie die Regeln einhalten und die Post den Brief befördert.

Der Nachrichtentyp als Umschlag Um einen Brief zu verschicken, müssen Sie sich an die Regeln für Umschlag und Porto halten. In Service Broker entsprechen sie dem Nachrichtentyp. Ein Service Broker-Nachrichtentyp kann den Wert `XML` oder `NONE` annehmen.

Beim Typ `XML` versucht SQL Server, vor der Verarbeitung die Gültigkeit des Dokuments zu prüfen. Alles, was Text verwenden kann, kann diesen Typ benutzen. Außerdem können Sie festlegen, dass er mit einem bestimmten XML-Schemadokument verarbeitet werden soll, um die Gültigkeit des Inhalts zu prüfen.

Der Nachrichtentyp wird beim Senden der Nachricht mit den T-SQL-Befehlen `SEND` und `RECEIVE` festgelegt. Um neue Typen zu erstellen, verwenden Sie den Befehl `CREATE MESSAGE TYPE`. Diesen Vorgang zeige ich Ihnen im Abschnitt *„Was Sie sich merken sollten“* am Ende des Kapitels.

Inhalte Sie können mit der Post ganze Bücher auf einmal verschicken, aber das ist bei Service Broker nicht unbedingt angebracht. Sie arbeiten mit einem Computersystem, das übliche Netzwerkleitungen benutzt, und daher ist es normalerweise am besten, die Nachricht in kleinere Portionen zu unterteilen.

Sie erstellen die Nachricht in Ihrem Ausgangsprogramm, dem sogenannten Initiator. Die meisten Dokumente werden als XML-Dateien verschickt.

Der Absender: Initiatoren und Verträge

Wenn Sie Post verschicken, müssen Sie die Nachricht zu Papier bringen, beide Adressen korrekt auf den Umschlag schreiben und das richtige Porto aufkleben. Der Postdienst garantiert, dass der Brief zugestellt wird, wenn Sie sich an die Regeln halten. Dies ist ein Vertrag, wie er auch im Service Broker vorkommt.

Der Benutzer oder ein Programm, der sogenannte Initiator, gibt die Informationen ein, die gesendet oder empfangen werden sollen. Das Programm führt eine gespeicherte Prozedur aus, die eine Nachricht erstellt und eine Konversation mit dem passenden Dienstprogramm von Service Broker einleitet, was ich demnächst erläutere.

Die Postlogistik: Physischer Transport, Protokolle und Bindung von Remoteservern

Obwohl der Vorgang größtenteils verdeckt abläuft, benutzen Sie dennoch eine Methode, um diese Daten vom Client auf den Server zu übertragen. Bei der Postlogistik fahren Angestellte mit Fahrzeugen oder gehen zu Fuß, um Briefkästen zu leeren oder Post auszutragen. In Service Broker erledigen physische Transportmittel (beispielsweise die Netzwerkinfrastruktur) und Protokolle (beispielsweise TCP/IP) den Transport von einer Stelle zur nächsten.

Das mag auf der Hand liegen, muss aber im Gesamtbild berücksichtigt werden. Würde die Postlogistik den Einsatz von Fahrzeugen ignorieren, würde möglicherweise die Routinewartung nicht durchgeführt oder würden keine Reservefahrzeuge eingeplant. Dasselbe gilt für Service Broker. Sie müssen daran denken, dass es in Ihrem System physische Bestandteile gibt, und Wartung und Sicherungen für einen Ausfall einplanen.

Sie können den gesamten Kommunikationsvorgang mit Zertifikaten schützen. Damit erreichen Sie eine hohe Verschlüsselungsstufe, sodass die Nachricht schwer zu entschlüsseln ist.

Das Postamt: Endpunkte, Warteschlangen und Aktivierungen

Ich erläutere Service Broker mithilfe der Postanalogie von Microsoft, habe mich aber vom Client zum Empfänger bewegt. Das war Absicht, damit Sie sehen, wie alles zusammenpasst, aber sowohl bei der Post als auch bei Service Broker läuft es nicht so. Bevor sich jemand hinsetzt und einen Brief schreibt, muss das Postamt gebaut werden. Genauso funktioniert es in Service Broker: Normalerweise errichten Sie zuerst den Kern und arbeiten von dort aus weiter.

Jedes Postamt hat Türen, die nur die Beschäftigten benutzen dürfen. In Service Broker heißen sie Endpunkte. Es sind direkte Netzwerkverbindungen zu einem Dienst. Die bereits erwähnten Gesprächsgruppen senden über diese Endpunkte Nachrichten zwischen den Diensten. Die meiste Zeit verwendet der Datenbankadministrator auf das Erstellen und Beobachten von Endpunkten.

Die Warteschlange von Service Broker entspricht einer einzelnen Ablage im Postamt. Sie ist das Datenlager für alle Nachrichten. Anders als im Postamt kann eine Nachricht aus einer Warteschlange an mehrere Stellen gesendet werden. Ich vermute, die Ähnlichkeit mit dem Postamt ist größer, als ich denke; manche Menschen bekommen nämlich große Mengen derselben unerwünschten Post.

Bei der Post ist ein Mitarbeiter dafür zuständig, an einer bestimmten Stelle bestimmte Arten von Post zu bearbeiten. In Service Broker wird jede Warteschlange mit einer gespeicherten Prozedur oder einem verwalteten Programm in der CLR-Schicht verknüpft: einem sogenannten Dienstprogramm. Wenn eine Nachricht bei der Warteschlange eingeht, wird das Dienstprogramm aktiviert und beginnt, sie zu verarbeiten. Die meiste Arbeit des Entwicklers gilt diesen gespeicherten Prozeduren. Sie entsprechen den Postmitarbeitern.

Hier zeigt sich ein weiterer Vorteil des Umstands, dass Service Broker direkt in der Datenbank ausgeführt wird. Diese Architektur macht die Installation mehrerer Dienste im Betriebssystem überflüssig. Da

die Warteschlange mit einer gespeicherten Prozedur verknüpft ist, „erwacht“ diese, sobald es Arbeit gibt, aber nicht vorher. Im Hintergrund muss nichts ausgeführt werden, was ständig auf Arbeit wartet.

Ein weiterer wichtiger Teil dieser Analogie besteht darin, dass ein Postmitarbeiter die Nachricht zwar dem Empfänger zustellen, sie aber auch zur weiteren Verarbeitung einem anderen Postmitarbeiter übergeben kann. In Service Broker können die gespeicherten Prozeduren genauso verfahren. Je nachdem, wie der Code lautet, kann die Nachricht verarbeitet und eine Abschlussmeldung gesendet oder zur weiteren Verarbeitung an eine andere Prozedur weitergeleitet werden.

Wenn die Verarbeitung abgeschlossen ist, wird die Konversation beendet. Das aufrufende Programm kann entweder den Status von Service Broker abrufen oder ohne Prüfung fortfahren.

7.2.2 Sicherheit

Anders als andere Datenbankanwendungen leitet Service Broker Nachrichten zwischen Systemen weiter. Deshalb sind Sie weniger mit einzelnen Benutzerkonten befasst. Dieser Bereich ähnelt eher der in Kapitel 4, *„Sicherheit“*, erläuterten Anwendungssicherheit, weil nur ein Konto eingesetzt wird, um im Auftrag des Benutzers auf das System zuzugreifen.

In Service Broker konzentriert sich die Sicherheit auf zwei Bereiche: Dialog und Transport. Dialoge haben mit Nachrichten zu tun, der Transport mit dem Netzwerk. Wenn sie geschützt werden, ist das System sicher.

Die Dialogsicherheit wird dadurch eingerichtet, dass in der Anwendungsdatenbank ein Benutzer und ein Zertifikat für ihn angelegt werden. Von dort aus können Sie den Benutzer und Service Broker mithilfe des Mechanismus der Dienstbindung verknüpfen. Die Anwendung setzt diesen Benutzer ein, um Daten zu senden und zu empfangen.

Für die Transportsicherheit richten Sie in der Datenbank `master` eine Serveranmeldung und ein Zertifikat ein und senden es an die aufrufenden Programme und den Datenbankadministrator der Anwendung. Das System verwendet diese Anmeldung zur Verarbeitung der Transaktionen.

7.2.3 Überwachung und Leistungsoptimierung

Um das System zu verwalten und zu überwachen, stehen Ihnen vier neue dynamische Verwaltungssichten und drei Leistungsobjekte und -indikatoren zur Verfügung.

Die Sichten sind in der folgenden Tabelle zusammengefasst:

`sys.dm_broker_activated_tasks`	Zeigt alle Aktivierungen gespeicherter Prozeduren.
`sys.dm_broker_connections`	Zeigt alle Service Broker-Netzwerkverbindungen.
`sys.dm_broker_forwarded_messages`	Zeigt die Nachrichten, die gerade weitergeleitet werden.
`sys.dm_broker_queue_monitors`	Zeigt den Warteschlangenmonitor, der die Aktivierung für eine Warteschlange verwaltet.

Tabelle 7.2 Dynamische Verwaltungssichten

Um den Leistungswert anzuzeigen, fügen Sie folgende Objekte und Indikatoren in Ihre Ablaufverfolgung ein:

SQLServer: Brokeraktivierung	Zeigt die Aktivierungen gespeicherter Prozeduren.
SQL Server: Broker-Statistik	Zeigt allgemeine Informationen über Service Broker.
SQLServer: Broker/DBM-Transport	Zeigt das Zusammenwirken von Service Broker und der Datenbankspiegelung.

Tabelle 7.3 Objekte und Indikatoren für die Ablaufverfolgung

Die beiden wichtigsten Optimierungsstrategien für Service Broker sind die Layoutplanung und die Optimierung der gespeicherten Prozeduren, die in der Anwendung als Dienstprogramme fungieren. Die richtige Planung der Infrastruktur bildet normalerweise den Grund für die Implementierung von Service Broker. Wenn Sie eine Lösung analysieren, sollten Sie grundsätzlich fragen, ob eine asynchrone Architektur möglich oder günstiger ist, und dann das entsprechende Layout umsetzen.

Wie bei den meisten Anwendungen betrifft der größte Teil der Optimierungsarbeit den T-SQL-Code, der in den gespeicherten Prozeduren ausgeführt wird. Zu den Strategien, auf die Sie die Entwickler aufmerksam machen sollten, zählt die Verwendung der Anweisung `WAITFOR`, die die gespeicherte Prozedur anweist, auf ein Ereignis zu warten, bevor sie den Code durchläuft. Diese Anweisung spielt in einer verbindungslosen Architektur eine große Rolle, weil die Nachricht sonst möglicherweise nicht vollständig ist, wenn das Dienstprogramm aktiviert wird.

7.3 Was Sie sich merken sollten

Sowohl bei Notification Services als auch bei Service Broker geht es mehr um Programmierkonstrukte als um die Verwaltung. Der Datenbankadministrator hat im täglichen Betrieb – abgesehen von den Konsequenzen für die Sicherheit und einigen zusätzlichen Leistungsobjekten – nicht viel Arbeit damit.

Ich war der Meinung, es könne sinnvoll sein, ein einfaches Beispiel für eine Service Broker-Anwendung durchzuarbeiten, weil es am ehesten mit gängigen T-SQL-Anweisungen umzusetzen ist. Eine eigene Anwendung zu erstellen trägt zum Verständnis des Vorgangs bei. Außerdem können Sie dieses einfache Programm sogar als Ausgangspunkt für eine komplexere Anwendung benutzen.

Ich werde die Kommentare auf das beschränken, was ich für wissenswert halte, weil ein Überblick über das Erstellen der Anwendung dem Verständnis dient. Um ihm folgen zu können, sollten Sie die Begriffe kennen, die ich im Verlauf des Kapitels erklärt habe. Wenn Sie an der Erweiterung des Systems interessiert sind, sollten Sie die Anwendung auf Ihrem Testsystem genauso erstellen. Anschließend überarbeiten Sie die T-SQL-Anweisungen, aus denen der Vorgang besteht, um das System zu erweitern.

7.3.1 Beispiel für Service Broker

In diesem Abschnitt führe ich ein einfaches Beispiel für die Aufnahme eines Mitarbeiters in eine Datenbank vor. Ein Personalsachbearbeiter muss in der Lage sein, einen Eintrag in das Mitarbeitersystem der Firma vorzunehmen. Da der Abteilungsleiter den Eintrag überprüfen möchte, bevor der Mitarbeiter eine dauerhafte ID bekommt, haben wir uns entschlossen, eine Anwendung zu schreiben, die den Eintrag in einer „Interimstabelle“ ablegt, bis er überprüft ist. Auf meinem Testsystem habe ich eine Datenbank `ServiceBrokerExample` mit einer Tabelle `Employee` angelegt. Diese Tabelle enthält eine Spalte `EmployeeInfo` mit der Spalteneinstellung `xml`, weil meine Anwendung dies erwartet. Für den Fall, dass Sie es auf Ihrem System ausprobieren wollen, folgt hier der entsprechende Code:

```
-- Datenbank anlegen

CREATE DATABASE ServiceBrokerExample

GO

-- Tabelle anlegen

USE [ServiceBrokerExample]

GO
```

```
CREATE TABLE [dbo].[Employee](
[EmployeeInfo] [xml] NULL
) ON [PRIMARY]
GO
```

Nachdem dies festgelegt ist, erstelle ich einen Nachrichtentyp für das System. Er soll überprüfen, ob meine Daten richtig im XML-Format vorliegen, wobei Sie in der Produktion häufig auf ein vollständiges XML-Schemadokument verweisen:

```
-- Nachrichtentyp erstellen
CREATE MESSAGE TYPE
[ServiceBroker/Example/Employee/AddEmployee]
VALIDATION = WELL_FORMED_XML
GO
```

Anschließend verfasse ich einen Vertrag für das System, den alle Parteien verwenden. Sie sehen, dass er den gerade erstellten Nachrichtentyp `AddEmployee` benutzt:

```
-- Vertrag verfassen
CREATE CONTRACT
[ServiceBroker/Example/Employee/AddEmployeeContract]
([ServiceBroker/Example/Employee/AddEmployee]
SENT BY INITIATOR
)
GO
```

Nun folgen meine „Postmitarbeiter", die Daten aus der Warteschlange lesen und in die Datenbank einfügen. Ich verwende eine XML-Konvertierungsfunktion, um die Daten zu übertragen und in die Datenbank zu schreiben. Sie besteht aus zwei gespeicherten Prozeduren: einer zum Einfügen und einer zum Überprüfen und Leeren der Warteschlange. Lassen Sie sich nicht von der Komplexität bremsen; gehen Sie die Abschnitte zeilenweise durch, um zu sehen, was geschieht:

```
-- Gespeicherte Prozedur zum Einfügen anlegen
CREATE PROCEDURE [dbo].[AddEmployee]
@MB xml
AS
INSERT INTO Employee(EmployeeInfo)
```

```
VALUES (@MB)
GO
```

Jetzt wird die gespeicherte Prozedur zum Lesen der Warteschlange erstellt:

```
-- Gespeicherte Prozedur zum Aktualisieren erstellen
CREATE PROCEDURE [dbo].[ProcessEmployee]
AS
BEGIN
BEGIN TRAN
DECLARE @CH uniqueidentifier
DECLARE @MB varbinary(max)
-- Entnimmt die Nachricht aus der Warteschlange
WAITFOR
(
RECEIVE TOP(1) @CH = conversation_handle, @MB =
message_body
FROM EmployeeQueue
),
TIMEOUT 1500
-- Nachricht verarbeiten
EXECUTE ProcessEmployee @MB
END CONVERSATION @CH
COMMIT TRAN
END
```

Nun muss ich das „Postamt“ einrichten, d.h. die Warteschlange, die die Daten aufnimmt. Beim Anlegen der Warteschlange weise ich das Dienstprogramm zu (in diesem Fall meine gespeicherte Prozedur), das die Schlange verarbeiten soll:

```
-- Warteschlange anlegen
CREATE QUEUE EmployeeQueue
WITH STATUS = ON,
ACTIVATION
(
```

```
PROCEDURE_NAME = ProcessEmployee,
MAX_QUEUE_READERS = 5,
EXECUTE AS SELF
)
GO
```

Ich habe es fast geschafft. Jetzt erstelle ich den Dienst, der auf die Anforderungen der Gespräche reagiert, und binde ihn an den bereits verfassten Vertrag:

```
-- Dienst erstellen
CREATE SERVICE AddEmployeeService
ON QUEUE [EmployeeQueue]
([ServiceBroker/Example/Employee/AddEmployeeContract])
GO
```

Das System ist fertig, und ich kann alle Objekte mit SQL Server Management Studio untersuchen. Da der Server für Service Broker-Konversationen bereit ist, kann ich ein vollständiges Beispielereignis einrichten. Ich sende nur ein kleines Stück des Codes, den ich normalerweise als XML-Dokument verwenden würde, weil sich der Code so leichter lesen lässt:

```
-- Dialog aufnehmen
DECLARE @CH uniqueidentifier
DECLARE @EmployeeName XML
SET @Employeename = '<name>Buck</name>'
BEGIN DIALOG CONVERSATION @CH
FROM SERVICE AddEmployeeService
TO SERVICE
'[ServiceBroker/Example/Employee/AddEmployeeService]'
ON CONTRACT
[ServiceBroker/Example/Employee/AddEmployeeContract];
SEND ON CONVERSATION @CH
MESSAGE TYPE
[ServiceBroker/Example/Employee/AddEmployee] (@EmployeeName)
GO
```

Um die Ergebnisse zu prüfen, frage ich die schon erwähnten dynamischen Verwaltungssichten und die Zieltabelle ab:

```
-- Ergebnisse einsehen

SELECT * FROM sys.dm_broker_activated_tasks

SELECT * FROM sys.dm_broker_connections

SELECT * FROM sys.dm_broker_forwarded_messages

SELECT * FROM sys.dm_broker_queue_monitors

GO

SELECT *

FROM

Employee

GO
```