Kapitel 1
Die Versionsgeschichte von SQL Server

In diesem Kapitel erfolgt zunächst ein kurzer geschichtlicher Überblick mit den wichtigsten Meilensteinen in der Geschichte von SQL Server. Anschließend stellen wir Ihnen die wichtigsten neuen Features der letzten Versionen von SQL Server vor. In den weiteren Kapiteln führen wir Sie dann in die Details dieser Features ein. Verschaffen Sie sich hier einen ersten Überblick.

SQL Server ist eine relationale Datenbank, die sich am Standard der aktuellen SQL-Version orientiert. SQL Server ging aus einer Zusammenarbeit der Firmen Microsoft und Sybase gegen Ende der 1980er Jahre hervor. Im Jahr 1989 kam die erste Version für OS/2 auf den Markt, ein von Microsoft und IBM entwickeltes Betriebssystem. Zu diesem Zeitpunkt war SQL Server kein eigenes Microsoft-Produkt, sondern entsprach dem *Sybase SQL Server* in der Version 4.0. Erst mit der Version 6.0 im Jahr 1995 kam eine eigenständige Weiterentwicklung von Microsoft auf den Markt, der die Version 6.5 im Jahr 1995 folgte, die jedoch von der Codebasis her immer noch dem Sybase-Produkt entsprach. Das änderte sich mit der Version 7.0. Jetzt hatte Microsoft eine neue, eigene Datenbank-Engine zuzüglich Codebasis entwickelt, mit der die Erfolgsgeschichte von Microsoft SQL Server begann.

Tabelle 1.1 zeigt die Meilensteine von SQL Server vom Zeitpunkt der Kooperation mit der Firma Sybase bis zum aktuellen Release von SQL Server.

Jahr	Version	Codename
1989	Version 1 für OS/2 (16 Bit – OS/2)	Filipi
1991	Version 1.1 (16 Bit – OS/2)	Pietro
1992	Version 4.2A für OS/2 (16 Bit – OS/2)	
1993	Version 4.2B (16 Bit – OS/2)	
1993	Version 4.21a (WinNT)	SQLNT

Tabelle 1.1 Historie der SQL-Server-Versionen

Jahr	Version	Codename
1995	Version 6.0	SQL95
1996	Version 6.5	Hydra
1999	Version 7.0/neue DB-Engine + Codebasis + OLAP Tools	Palato mania
2000	Version 8.0/SQL Server 2000	Siloh
2003	Version 2000 64 Bit Edition	Liberty
2005	Version 9.0/SQL Server 2005 Unterstützung für .NET	Yukon
2008	Version 10.0/SQL Server 2008	Katmai
2010	Azure SQL Database (initial release)	Cloud Database oder CloudDB
2010	Version 10.5/SQL Server 2008 R2	Kilimanjaro (kurz KJ)
2012	Version 11.0.2100.60/SQL Server 2012	Denali
2014	Version 12.0/SQL Server 2014	Hekaton
2016	Version 13.0/SQL Server 2016	
2017	Version 14.0/SQL Server 2017	vNext
2019	Version 15.0/SQL Server 2019	Aris

Tabelle 1.1 Historie der SQL-Server-Versionen (Forts.)

1.1 Entwicklung bis Microsoft SQL Server 2005

Seit der Version 2000 von Microsoft SQL Server sind standardmäßig eine Volltextsuche und OLAP-Funktionalitäten integriert. Diese OLAP-Funktionalitäten werden seit der Version 2005 *SSAS* – oder besser *SQL Server Analysis Services* – genannt. Zum Lieferumfang gehörte ein grafisches Tool für die Datenbankverwaltung und -programmierung. Der Enterprise Manager diente der vollständigen grafischen Verwaltung und der Query-Analyzer, der auch Funktionen zur Programmierung und Optimierung der Datenbank mitbringt, zur codebasierten Verwaltung.

Mit SQL Server 2005 wurden zahlreiche Neuerungen eingeführt. Diese betrafen sowohl Administratoren als auch Entwickler. Zu den Neuerungen für Administratoren zählten z. B. die Datenbankspiegelung, der Protokollversand, die Integration Services,

Sicherheitserweiterungen und die neuen Services. Gerade die neuen Services – *Integration Services* (*SSIS*), *Reporting Services* (*SSRS*) und *Analysis Services* (*SSAS*) – waren es, die SQL Server einen erheblichen Mehrwert verliehen und somit dafür sorgten, dass SQL Server das am schnellsten wachsende Datenbankprodukt am Markt wurde.

In vielen Seminaren haben wir immer wieder erlebt, wie begeistert Administratoren und Entwickler von den neuen Möglichkeiten von SQL Server waren. Es bereitet uns immer wieder große Freude, zu sehen, wie begeistert die Seminarteilnehmer sind, sobald sie die Möglichkeiten von SQL Server entdecken oder darauf aufmerksam gemacht werden. Technologien und Neuerungen in einem Seminar zu vermitteln, ist sicherlich die eine Seite; diese dann jedoch im weiteren Verlauf gemeinsam mit den Unternehmen umzusetzen, ist immer wieder eine weitere sehr schöne Aufgabe. Es macht uns immer glücklich, wenn wir sehen, wie zufrieden Administratoren oder Entwickler sind, die neue Lösungen und Technologien einführen, um damit ihre Probleme beheben zu können. SQL Server 2005 war sicherlich ein Meilenstein in der Geschichte dieses Produktes.

Hinzu kam die Integration der *Common Language Runtime* (*CLR*), die die Entwicklung von .NET-Code in SQL Server erlaubte. Viele Entwickler waren hocherfreut über die Integration von Visual Studio und die Möglichkeiten, professionelles Debugging durchzuführen. Allein die Möglichkeit, auf die sehr umfangreiche Klassenbibliothek des *.NET Frameworks* zuzugreifen, war ein echter Höhepunkt. Ein neues einheitliches Verwaltungstool mit dem Namen *SSMS – SQL Server Management Studio* – wurde eingeführt, mit dem es jetzt möglich war, GUI- und codebasiert innerhalb einer Umgebung zu arbeiten. Somit wurden der Query-Analyzer und der Enterprise Manager abgelöst. Visual Studio erlaubte es zudem, eine Solution anzulegen und mit gängigen Quellcode-Verwaltungssystemen zu arbeiten. Eine vollständige XML-Unterstützung, bei der nicht nur der Datentyp XML existierte, war ebenfalls Bestandteil der Version 2005.

Seit der Version 7 stellt Microsoft eine kostenlose Variante von SQL Server zur Verfügung. Diese wurde bis zur Version 2000 MSDE (Microsoft Desktop Engine) genannt und mit der Version 2008 in *Microsoft SQL Server Express Edition* umbenannt. Die SQL Server Express Edition hat wesentlich weniger Einschränkungen als die MSDE. Es gibt keine Workload-Beschränkung mehr, und neben dem *Management Studio Express* ist der Assistent zum Importieren und Exportieren von Datenbanken enthalten. Ein weiteres Ziel von Microsoft war es, Entwicklern die Möglichkeit zu geben, nicht auf Basis von Microsoft Access zu entwickeln, sondern gleich den Code auf Basis von SQL Server aufzubauen. Seitdem müssen keine Änderungen am Code vorgenommen werden, falls es zu einer Skalierung kommt. Es ist dann lediglich notwendig, eine SQL-Server-Lizenz zu erwerben. Natürlich unterliegt die kostenlose Version einigen Einschränkungen. In Kapitel 3, »Die SQL-Server-Editionen im Überblick«, geben wir Ih-

nen eine Übersicht über die aktuellen Versionen von SQL Server und deren Leistungsumfang.

1.2 Von Microsoft SQL Server 2008 zu SQL Server 2014

Mit der Version 2008 erfolgten zahlreiche Verbesserungen, die sich Kunden im Zusammenhang mit SQL Server 2005 gewünscht hatten. So wurden Features wie die transparente Datenbankverschlüsselung, die richtlinienbasierte Verwaltung, der Datensammler, die Ressourcenverwaltung, Change Data Capture und Change Tracking eingeführt. Des Weiteren hat sich Microsoft auf die Fahne geschrieben, das Thema *Business Intelligence* (BI) weiter in den Unternehmen zu etablieren und es für jeden möglichst einfach zugänglich zu machen. Dieses Ziel wurde mit SQL Server 2008 R2 weiterverfolgt, und Themen wie Self-Service-BI und PowerPivot stehen heute auf der Tagesordnung.

Jeder neue SQL Server hatte seinen eigenen Schwerpunkt, so auch die Version 2012 – *Mission Critical* lautete die Überschrift, unter der SQL Server 2012 eingeführt wurde. Seine Berechtigung erlangt der neue SQL Server 2012 zum einen sicherlich durch die stetig wachsende Datenflut in den Unternehmen und zum anderen durch mobile Dienste und das Thema Cloud. Das alles führt zu einer stetigen Weiterentwicklung auch eines erfolgreichen Produktes. Keine leichte Aufgabe für SQL Server 2012. Wir möchten Ihnen im Folgenden einen kurzen Überblick über die neuen Features von SQL Server 2012 geben. Dieser Überblick soll Ihnen dabei helfen, die einzelnen Features und ihre Vorteile zu verstehen.

1.2.1 SQL Server 2012: Hochverfügbarkeit

Die *Hochverfügbarkeit* hat Microsoft seit der Einführung der Datenbankspiegelung bzw. dem Protokollversand mit SQL Server 2005 stetig ausgebaut und verbessert. Das ist auch sehr gut nachvollziehbar, wenn man sich die zunehmende Anzahl von Microsoft-SQL-Server-Installationen am Markt ansieht. Wir denken hier nicht nur allein an Firmen wie SAP, sondern auch an Produkte aus dem eigenen Haus, wie SharePoint. Die Datenbanken entwickeln sich immer mehr zu unternehmenskritischen Anwendungen, im Falle von SharePoint zur zentralen Dokumentenablage, zur Suche nach Informationen, zum Intranet oder zu Services wie Microsoft Excel, InfoPath und Business Connectivity.

Wir betreuen in Projekten bzw. Seminaren sehr viele Kunden in diesem Produktumfeld, und es wurde hier schnell deutlich, was geschieht, wenn eine Instanz von SQL Server ausfällt. Nun ist der Microsoft SQL Server mit Sicherheit ein ausgereiftes Produkt, jedoch gibt das allein keine Garantie dafür, dass keine Ausfälle auftreten. Nicht zuletzt können auch Menschen Fehler machen, sei es beim Thema Patchen, der Entwicklung von Softwarelösungen oder sonstigen Tests. Die bisherigen Möglichkeiten von SQL Server selbst erlaubten die Datenbankspiegelung oder den Protokollversand, beides auf Datenbankebene. Wer mehr wollte, z. B. Ausfallsicherheit für die gesamte Instanz von SQL Server, musste auf einen Failover-Cluster zurückgreifen.

1.2.2 SQL Server 2012: SQL Server AlwaysOn

Mit der Einführung von SQL Server 2012 gibt es eine neue Hochverfügbarkeitslösung für SQL Server in der Enterprise Edition. Diese nennt sich *AlwaysOn* und sorgt für eine verbesserte Hochverfügbarkeit, ohne dass ein gemeinsamer Speicher wie beim Failover-Cluster notwendig ist. AlwaysOn setzt auf dem Windows-Cluster-Dienst mit bis zu fünf Knoten auf und bietet die Möglichkeit, mehrere sekundäre Datenbanken vorzuhalten. Datenbanken auf SQL Server 2012 können so redundant über mehrere Rechenzentren verteilt laufen. Dabei ist ein lesender Zugriff auf die Replikate möglich. Diese Technologie wird bereits von der Microsoft-Azure-Plattform eingesetzt und sorgt dort dafür, dass, wenn ein Knoten (Instanz von SQL Server) ausfällt, ein anderer Knoten den Dienst übernimmt. Wie bereits in der Vergangenheit bei der Datenbankspiegelung gibt es für die Synchronisation der Transaktionen zwei Verfahren – synchron und asynchron. SQL Server 2019 utnerstützt die Datenbankspiegelung zwar noch, allerdings hat Microsoft diese als veraltet gekennzeichnet und bereits angekündigt, sie in zukünftigen Versionen von SQL Server nicht mehr unterstützen zu wollen. Darüber hinaus kann unter AlwaysOn beim Failover zwischen automatisch oder manuell entschieden werden.

In der Praxis sieht es so aus, dass bei der Konfiguration von AlwaysOn Verfügbarkeitsgruppen unter einem virtuellen Namen angelegt werden, die wiederum eine IP-Adresse per DHCP oder statisch erhalten. Diese Verfügbarkeitsgruppen enthalten eine oder mehrere Datenbanken, die dann auf einen festgelegten Knoten synchronisiert werden.

Hier zeigt sich schon eine Stärke dieser neuen Technologie: Es ist damit möglich, mehrere Datenbanken auf einen Knoten als Gruppe zu synchronisieren. Wir denken hier an all die Anwendungen, die aus mehr als einer Datenbank bestehen. Ein weiterer wesentlicher Vorteil von AlwaysOn besteht darin, dass auf die sekundären Knoten lesend zugegriffen werden kann, was die Verfügbarkeit erhöht und eine Skalierung ermöglicht.

1.2.3 SQL Server 2012: Skalierbarkeit und Performance

Auch im Bereich der Skalierbarkeit und Performance wartet SQL Server 2012 mit neuen Features auf. Bei stetig zunehmendem Datenvolumen besteht die Notwendigkeit, ein Produkt immer weiter hinsichtlich der Performance und Skalierbarkeit zu

überarbeiten. Seit SQL Server 2005 wurden viele neue Features wie die Partitionierung von Tabellen, abdeckende Indizes, XML-Indizes und vieles mehr eingeführt. Im aktuellen Release wurden neue Features bzw. Verbesserungen an vorhandenen Features vorgenommen.

1.2.4 SQL Server 2012: Columnstore-Indizes

Neu in SQL Server 2012 sind die *Columnstore-Indizes*. Sie bieten eine einfache und zugleich sehr wirksame Methode zur Verbesserung der Abfrageleistung über sehr große Datenmengen. Die spaltenorientierte Speicherung von Indizes, zusammen mit der In-Memory-Verarbeitung, ist sehr effizient, speziell in Data-Warehouse-Lösungen. Damit lassen sich Abfragezeiten um ein Vielfaches verringern und Speicherplatz sparen.

1.2.5 SQL Server 2012: FileTable

FileStream bietet seit SQL Server 2008 die Möglichkeit, BLOBs wie Dateien, Audio, Video etc. über SQL Server im Dateisystem abzulegen. Dabei wird für die Instanz von SQL Server eine Freigabe erstellt, mit deren Hilfe die BLOBs in das Dateisystem ausgelagert werden. Die Kontrolle erfolgt über die jeweilige API bzw. über T-SQL und SQL Server. Der Vorteil dieser Lösung besteht darin, dass die Datenbanken klein bleiben und die Daten im Dateisystem in Bezug auf die restlichen Daten in den Tabellen synchron gehalten werden. Der Nachteil dieser Lösung ist, dass der Zugriff nur über eine API oder T-SQL erfolgen kann, nicht jedoch einfach über das Dateisystem aus irgendeiner Anwendung heraus, wie z. B. Photoshop. Genau hier setzt *FileTable* an. FileTable ist eine Tabelle mit einem fixen Schema. Sie kann also nicht erweitert werden, um z. B. zusätzliche Metadaten darin zu speichern. Diese Tabelle speichert Dokumente, die nun sowohl über T-SQL als auch ganz normal über das Dateisystem verwaltet werden können. Der Umgang und die Verwaltung sind einfach umzusetzen. FileTables lassen sich auch per Volltextsuche und statistischer Semantik indizieren.

1.2.6 SQL Server 2012: Volltextsuche

In der Volltextsuche wurden einige Verbesserungen eingeführt. Dazu zählt die Eigenschaftssuche, die z. B. Microsoft-Office-Dokumenten nach Metadaten wie Autor oder Titel durchsucht. Neben den genannten Anpassungen in der Volltextsuche wurde an der Performance und Skalierbarkeit gearbeitet. Das Ziel ist es, mindestens 100 Millionen Dokumente in einem Volltextindex zu unterstützen. Neu hinzugekommen ist eine semantische Suche, bei der typische Fragestellungen beantwortet werden sollen, wie z. B. welche Dokumente ähneln diesem Dokument. Die semantische Suche ist eine Erweiterung der Volltextsuche.

1.2.7 SQL Server 2012: Benutzerdefinierte Serverrollen

Seit SQL Server 2012 können eigene Serverrollen angelegt werden. Bis dahin war das lediglich für Datenbankrollen möglich. Der Vorteil der benutzerdefinierten Serverrollen wird schnell ersichtlich, wenn eine größere Anzahl von Personen innerhalb der Instanz von SQL Server berechtigt werden soll. Ein Beispiel ist hier das Anzeigen von Sperrinformationen in SQL Server. Dazu muss den Benutzern auf der SQL-Server-Instanz das entsprechende Recht zugewiesen werden. Eine selbst definierte Rolle ist hier in Zukunft sicher sehr hilfreich. Natürlich eignen sich Serverrollen auch dahingehend, dass bei mehreren Administratoren unterschiedliche, individuell abgestimmte Berechtigungsebenen für diese geschaffen werden können.

1.2.8 SQL Server 2012: SQL Server 2012 – Contained Database

Wer vor SQL Server 2012 einem Windows-Benutzer bzw. SQL-Server-Benutzer Zugriff auf eine Datenbank gewähren wollte, musste auf der Instanz von SQL Server ein Login anlegen. Damit war der Benutzer im Besitz des Connect-Rechts und konnte eine Verbindung zu SQL Server herstellen. Wenn nun im nächsten Schritt der Benutzer Zugriff auf eine Datenbank benötigte, musste in der jeweiligen Datenbank ein Datenbankbenutzer erstellt und dem Login zugewiesen werden.

Es waren also mehrere Schritte notwendig. Zum einen musste ein Login auf der Instanz angelegt werden, zum anderen die Zuweisung *Datenbankbenutzer* in der Datenbank erfolgen. Letztendlich mussten die jeweiligen Rechte innerhalb der Datenbank dem Datenbankbenutzer zugeordnet werden. Wurde diese Datenbank gesichert und auf einer anderen Instanz wiederhergestellt, existierte zwar der Datenbankbenutzer, der der Datenbank zugewiesen wurde, das Login fehlte jedoch, da es sich nicht in der Datenbank befand, sondern in der Master-Datenbank der ursprünglichen SQL-Server-Instanz.

Mit SQL Server 2012 führte Microsoft das Feature *Contained Database* ein. Dieses ermöglicht, eine Datenbank anzulegen, bei der die Benutzerinformationen innerhalb der Datenbank komplett abgelegt sind. Somit wird für diesen Benutzer kein Login mehr auf der Instanz von SQL Server benötigt, sondern der Benutzer mit Typ, Kennwort, Standardsprache und Standardschema innerhalb der Contained Database angelegt. Eine Verbindung zur Master-Datenbank entfällt nun, und beim Login muss die Datenbank angegeben werden.

Damit wird das Verschieben von Datenbanken in einer Testumgebung, Microsoft Azure SQL-Datenbank oder innerhalb von Hochverfügbarkeitslösungen wesentlich vereinfacht.

1.2.9 SQL Server 2012: Distributed Replay

Beim Stichwort *Replay* fällt dem einen oder anderen von Ihnen sicherlich sofort der SQL Server Profiler ein. Dieser bietet die Möglichkeit, Ablaufverfolgungen durchzuführen und diese dann zu einem späteren Zeitpunkt auf einer anderen Instanz von SQL Server auszuführen. Diese Funktion ist bereits seit SQL Server 2012 als veraltet gekennzeichnet. Da es immer etwas dauert, bis eine solche Technologie abgeschaltet wird, können Sie diese zwar noch einsetzen, allerding sollten Sie sich darauf einstellen, dass zukünftige Versionen von SQL Server keine Unterstützung mehr anbieten werden. Ausgenommen ist die Ablaufverfolgung von SSAS – SQL Server Analysis Services (OLAP). Weitere Hinweise dazu finden Sie unter dem folgenden Link: *http://msdn.microsoft.com/de-de/library/ms181091.aspx*.

Mit dem SQL Server Profiler kann lediglich eine einzige Arbeitsauslastung auf einem einzelnen Computer wiedergegeben werden. Weil jedoch in komplexen Umgebungen oft mehrere Serverinstanzen zum Einsatz kommen und eine Skalierung absolut notwendig ist, wurde zur Überwindung dieser Grenzen *Distributed Replay* einführt. Mit Distributed Replay können Sie eine Arbeitsauslastung von mehreren Rechnern wiedergeben. Damit kann die Arbeitsauslastung besser simuliert werden.

1.2.10 SQL Server 2012: SQL-Server-Audit-Erweiterungen

SQL Server Audit auf Serverebene ist seit SQL Server 2012 in allen Editionen von SQL Server verfügbar. Neue Funktionen zum Filtern ermöglichen eine bessere Handhabung bei der Suche nach bestimmten Events.

1.2.11 SQL Server 2012: Management Pack für Hochverfügbarkeit

Mit dem System-Center-Überwachungspaket für SQL Server hat Microsoft ein Management Pack veröffentlicht, mit dem sich die Eigenschaften und der Status von Verfügbarkeitsgruppen überwachen lassen. Damit haben Sie als Administrator die Möglichkeit, eine Hochverfügbarkeitslösung mit AlwaysOn über das Management Pack im System Center Operation Manager zu überwachen.

1.2.12 SQL Server 2012: Windows Server Core

SQL Server 2012 wird von *Windows Server Core* unterstützt. Damit sind Szenarien denkbar, bei denen sich der Patch-Aufwand reduziert. Diese Möglichkeit begünstigt erhöhte Sicherheit und Reduzierung des Verwaltungsaufwands. Wenn Sie in Zukunft SQL Server betreiben möchten, müssen Sie das daher nicht mehr zwingend in einer GUI-basierten Umgebung tun.

1.2.13 SQL Server 2012: PHP-Treiber

Microsoft stellt einen eigenen PHP-Treiber zur Verfügung, mit dem auf SQL Server seit der Version 2012 zugegriffen werden kann. Ausführliche Informationen hierzu bietet Ihnen die Seite *http://www.microsoft.com/en-us/download/details.aspx?id=20098*.

1.2.14 SQL Server 2012: LocalDB-Laufzeitumgebung

SQL Server LocalDB ist ein Ausführungsmodus der SQL Server Express Edition. Die Installation erfolgt über eine MSI-Datei mit dem Namen *SqlLocalDB.msi*. Hauptzielgruppe sind Entwickler, die auf der Grundlage von SqlLocalDB Anwendungen entwickeln, um diese dann in Zielumgebungen bereitzustellen. Das Hilfsprogramm *SqlLocalDB.exe* dient zur Verwaltung, umfangreiche Konfigurationsaufgaben entfallen, und mit den Developer Tools kann eine einfache Bereitstellung erfolgen.

1.2.15 SQL Server 2012: SQL Server Data Tools (SSDT)

Microsoft hat mit den *SQL Server Data Tools* und der Einführung eines deklarativen Modells in Visual Studio die Möglichkeit geschaffen, noch verzahnter mit Datenbanken aus Visual Studio heraus zu arbeiten. Datenbanken können entwickelt werden, und ein Debuggen ist möglich, gleichgültig, ob Sie im Projektmodus (offline) arbeiten oder im Onlinemodus mit der Datenbank verbunden sind. Hier sei auf die Importoptionen hingewiesen: Sie können eine Datenbank in Visual Studio in ein neu angelegtes Datenbankprojekt aus einem SQL-Skript, einer Datenebenenanwendung oder einer existierenden Datenbank importieren.

Nach dem Start der SQL Server Data Tools wird zunächst ein neues Datenbankprojekt angelegt. Wie z. B. bei der Entwicklung unter SharePoint 2010 gewohnt, lässt sich ein neues Element im Projektmappen-Explorer über HINZUFÜGEN hinzufügen.

Aufgrund des deklarativen Modells stehen jetzt Datenbankobjekte wie Tabellen, Benutzer, Dateigruppen, Rollen, Trigger und Schlüssel zur Verfügung. Die Objekte werden dann in Form des jeweiligen SQL-Skripts hinterlegt, und das Ganze lässt sich auch veröffentlichen. Die Datenbank mit Tabelle inklusive des Logins befindet sich im Anschluss auf dem SQL Server.

Um es an dieser Stelle kompakt zu halten, soll dieses kleine Beispiel genügen, um Ihnen einen kurzen Überblick darüber zu geben, was sich hinter den SQL Server Data Tools verbirgt. Letztendlich wurde die Integration in Visual Studio, die mit SQL Server 2005 begann, weiter vorangetrieben. Tabellendesign, Erzeugung des Codes, Codenavigation, IntelliSense, SQL-Server-Objekt-Explorer, Datenbankverwaltung und Entwurf – all das ist mit den SQL Server Data Tools möglich. Es befinden sich hier auch die

klassischen BI-Projektvorlagen (SSRS, SSIS, SSAS), die in der Vergangenheit unter dem SQL Server Business Intelligence Development Studio genutzt wurden.

Zusammenfassend lässt sich festhalten, dass mit den SQL Server Data Tools auf einfache Weise neue Datenbanken erstellt, verwaltet und ausgerollt werden können. Als Entwickler und Administratoren werden Sie sich gleichermaßen freuen, die SQL Server Data Tools zu verwenden.

1.2.16 SQL Server 2012: Data Quality Services

Um die Qualität von Daten zu verbessern, hat Microsoft die *Data Quality Services* eingeführt. Korrektur, Vervollständigung und Vermeidung von Redundanzen zur Sicherstellung von Datenqualität ist das Ziel, das mithilfe von Data Quality Services erreicht werden soll.

1.2.17 SQL Server 2012: PowerView – Report- und Analysetool

Schaut man sich an, was im SQL-Server-2012-BI-Bereich hinzugekommen ist, fällt zunächst das Reporting- und Analysetool *PowerView* auf, das unter dem Codenamen *Crescent* geführt wurde. PowerView, eine Funktion in Verbindung mit den SQL Server 2012 Reporting Services für Microsoft SharePoint Server 2010 in der Enterprise Edition, ist für die interaktive Durchsuchung, Visualisierung und Darstellung von Daten geeignet.

1.2.18 SQL Server 2012: Reporting als SharePoint Shared Service

Die zu SQL Server erhältlichen *Reporting Services* können direkt in einen SharePoint-Server integriert werden. In der Enterprise Edition von SQL Server enthalten sie die PowerView-Funktion in Anlehnung an das Analysewerkzeug PowerPivot für Microsoft Excel.

1.2.19 SQL Server 2012: SSIS-Bereitstellung von Projekten und Paketen

SSIS-Projekte können auf dem Integration-Services-Server bereitgestellt werden. Dazu muss ein SSIS-Katalog (Datenbank mit dem Namen SSISDB) angelegt werden. Das ermöglicht die Bereitstellung im Projektmodus. Der Katalog ist der zentrale Punkt zur Bereitstellung, Konfiguration und Paketverwaltung mit Parametern. Darüber hinaus besteht die Möglichkeit der Auswahl zwischen einer Bereitstellung im klassischen Projektmodus oder im Paketmodus.

1.2.20 SQL Server 2012: Tabellarische Projekte in den SQL Server Data Tools

Mithilfe der *SQL Server Data Tools* können tabellarische Projekte angelegt werden. Dafür existieren diese Projektvorlagen:

- Analysis-Services-Projekt für tabellarische Modelle
- vom Server importieren
- aus PowerPivot importieren

Mit ihrer Hilfe können Projekte erstellt werden. Ein Tabellenmodelldesigner wurde in die SQL Server Data Tools integriert und kann aus Visual Studio heraus genutzt werden. Darin stehen zwei Ansichten zur Verfügung: die Diagrammansicht und die Datensicht.

Wird ein Projekt als Tabellenmodell mithilfe der SQL Server Data Tools angelegt, existiert im Hintergrund eine Arbeitsbereichsdatenbank. Diese Arbeitsbereichsdatenbank befindet sich im Arbeitsspeicher auf der Analysis-Services-Instanz, die im tabellarischen Modus ausgeführt wird. Jedes tabellarische Projekt verfügt über eine eigene Arbeitsbereichsdatenbank. Mithilfe des SQL Server Management Studios kann diese auf der Analysis-Services-Server-Instanz angezeigt werden. Das xVelocity-Modul für Datenanalyse im Arbeitsspeicher (*VertiPag*) ist die technologische Grundlage. Im Gegensatz zur Vorgängerversion, bei der das Modul nur in Verbindung mit SharePoint genutzt werden konnte, steht es jetzt auch völlig eigenständig für Analysis Services zur Verfügung.

1.2.21 SQL Server 2014: Verbesserungen der Skalierbarkeit, Leistung und Performance

Die Verwaltung des Arbeitsspeichers je Prozessorkern und auch der Kerne untereinander wurde verbessert und mithilfe dynamischer Partitionierung auch ausfallsicherer gemacht.

Alte Speichergrenzen wurden nach oben verbessert, und die Synchronisierung von Daten wurde beschleunigt. Zusätzlich ist der Umgang mit Geo-Daten deutlich schneller geworden.

1.2.22 SQL Server 2014: Support und Diagnose erfahren eine deutliche Vereinfachung

Auch in diesem Bereich gibt es neue Funktionen – zum Beispiel das Erstellen einer Klon-Datenbank mit *DBCC*-Befehl oder aussagekräftigere Fehlermeldungen, wenn die TempDB Probleme hat; die *Instant File Initialization* wird protokolliert; inkrementelle Statistiken erhalten eine eigene Management-View; UTF-8 wird beim Massenim-

port besser unterstützt. Außerdem gibt es Verbesserungen rund um Performance und Diagnose.

Diese Version hat keine bahnbrechenden Features erhalten, dafür gibt es viele Verbesserungen für diejenigen, die damit arbeiten.

1.3 SQL Server 2016 – wichtige Neuerungen im Überblick

Bisher hatte jede neue Version von SQL Server ihren eigenen Schwerpunkt. Für SQL Server 2016 heißt dieser *Parallel Data Warehouse*. Er unterstützt Sie dabei, intelligente, unternehmenskritische Anwendungen auf einer hybriden, aber dennoch skalierbaren Plattform anzubieten. Alles, was Sie dazu benötigen, ist bereits integriert und muss nicht teuer hinzugekauft werden: In-Memory-Funktionalität, erweiterte Sicherheit und Datenanalyse sind bereits in der Datenbank. Hadoop- und Cloudintegration, R-Analyse und mehr gehören ebenalls schon zur Ausstattung.

1.3.1 SQL Server 2016: Verbesserungen des Datenbankmoduls

Die Version 2016 von SQL Server bringt einige Verbesserungen des Datenbankmoduls mit sich:

- Schon während der Installation können Sie mehrere TempDB-Dateien anlegen, was bei aktueller Hardware auf jeden Fall eine gute Idee ist.
- Der neue *Query Store* (Abfragespeicher) speichert Abfragen, Ausführungspläne und Leistungsmetriken. Dadurch ist es einfacher, Performanceprobleme zu finden und zu beheben. Ein neues Dashboard zeigt Langläuferabfragen und RAM- oder CPU-Fresser.
- In temporalen Tabellen wird der Verlauf von Datenänderungen mit Datum und Uhrzeit erfasst.
- Die neue *JSON-Integration* unterstützt Import, Export, Analyse und Speichervorgänge.
- Das neue *PolyBase*-Abfragemodul integriert SQL Server mit externen Daten in Hadoop oder Azure Blob Storage. So können Sie Daten importieren und exportieren sowie Abfragen quer über die integrierte Landschaft durchführen, so als wäre alles in Tabellen der lokalen Instanz enthalten.
- Die *Stretch-Database*-Funktion bietet die Möglichkeit, lokale Datenbanken dynamisch und sicher in einer Azure-SQL-Datenbank in der Cloud zu archivieren.
- *In-Memory-OLTP* wurde erweitert und unterstützt jetzt weitere Einschränkungen, wie z. B. `FOREIGN KEY`, `UNIQUE` und `CHECK` sowie die gespeicherten Prozeduren `OR`, `NOT`, `SELECT DISTINCT`, `OUTER JOIN` und Unterabfragen in `SELECT`. Dabei wurde die Größenbeschränkung von 256 GB auf 2 TB angehoben. Zusätzlich bietet es Erweiterungen des Columnstore-Index für die Sortierung und unterstützt die AlwaysOn-Verfügbarkeitsgruppen.
- *Always Encrypted* ist ein Feature, das die Daten permanent verschlüsselt hält, und nur die Anwendung, die den Schlüssel hält, kann auf die sensiblen Daten zugreifen. SQL Server selbst erhält niemals den Schlüssel.
- *Dynamische Datenmaskierung* (*Dynamic Data Masking*) kann dafür eingesetzt werden, die Daten zu maskieren, so dass User nur die Daten sehen können, die auch für sie gedacht sind.
- Sicherheit auf Zeilenebene (*Row-Level Security*) beschränkt den Zugriff auf Daten schon im Datenbank-Engine-Modul. Dadurch können nur berechtigte User die für sie vorgesehenen Daten einsehen.

1.3.2 SQL Server 2016: Analysis Services (SSAS)

Bei den SQL Server 2016 Analysis Services wurden die Leistung sowie die Möglichkeiten der Verwaltung, die Filterung und weitere Funktionalitäten für den Umgang mit tabellenbasierten Datenbanken verbessert.

- Die *SQL Server R Services* integrieren die für statistische Analyse verwendete, recht mächtige Programmiersprache *R* in SQL Server.
- Die Datenbank-Konsistenzprüfung (*Database Consistency Checker*, DBCC), die beschädigte Daten erkennt und dadurch entstehende Probleme erkennt, wird jetzt intern ausgeführt.
- Die Live-Abfrage externer Daten, die direkte Abfrage (*Direct Query*), unterstützt weitere Datenquellen, einschließlich Azure, Oracle und Teradata.
- Es gibt eine Vielzahl neuer DAX-Funktionen (*Data Access Expressions*).
- *Microsoft.AnalysisServices.Tabular* ist der neue Namespace, der Instanzen und Modelle im tabellarischen Modus verwaltet.
- Durch die Überarbeitung erhalten die *Analysis Services Management Objects* (*AMO*) jetzt eine zweite Assembly: *Microsoft.AnalysisServices.Core.dll*.

1.3.3 SQL Server 2016: Integration Services (SSIS)

Die Änderungen an den SSIS umfassen:

- *AlwaysOn-Verfügbarkeitsgruppen* werden jetzt unterstützt.
- Pakete können inkrementell bereitgestellt werden (*Incremental Package Deployment*).
- *Always Encrypted* wird auch von den Integration Services unterstützt.

- Auf Datenbankebene gibt es die neue Rolle *ssis_logreader*.
- Der Protokolliergrad kann jetzt auch vom Benutzer definiert werden.
- Im Datenfluss können Spaltennamen für Fehler definiert werden (*Column Names for Errors*).
- Neue Konnektoren wurden implementiert (*New Connectors*).
- Neu ist auch die Unterstützung des *Hadoop-Dateisystems* (*HDFS*).

1.3.4 SQL Server 2016: Master Data Services (MDS)

- Abgeleitete Hierarchien (*Derived Hierarchy Improvements*) einschließlich der Unterstützung rekursiver m:n-Beziehungen wurden verbessert.
- Das Filtern nach domänenbasierten Attributen (*Domain-Based Attributes*) sowie benutzerdefinierte Indizes (*Custom Indexes*) zur Verbesserung der Abfrageleistung sind neu.
- Jetzt können Entitäten zur gemeinsamen Nutzung von Entitäten in verschiedenen Modellen synchronisiert werden (*Entity Syncing*).
- Genehmigungsworkflows können über Changesets gemanagt werden, und neue Genehmigungslevel (*Permission Levels*) wurden für mehr Sicherheit implementiert.
- Zu guter Letzt gab es noch eine Neugestaltung der Geschäftsregelverwaltung (*Business Rules Management*).

1.3.5 SQL Server 2016: Reporting Services (SSRS)

Die SQL Server Reporting Services der Version 2016 erfuhren eine sehr gründliche Überarbeitung.

Neu sind z. B. ein webbasiertes Berichtportal (*Web Report Portal*) mit KPI-Funktion, ein Publisher für mobile Berichte (*Mobile Report Publisher*), ein Modul zum Rendern von Berichten (*Rendering Engine*), das HTML5 unterstützt, und – nicht zu vergessen – die neuen Diagrammtypen (englisch: *Chart Types*) Treemap und Sunburst.

1.3.6 Das neue MS SQL Server Management Studio (SSMS)

Um der Geschwindigkeit der Weiterentwicklung Rechnung zu tragen, hat Microsoft das wichtigste Verwaltungstool, das *SQL Server Management Studio*, aus dem Installationspaket von SQL Server 2016 herausgelöst. Ab der Version 2016 ist das Management Studio ein eigenständiges, kostenfrei nutzbares Softwarepaket. So ist es möglich, Updates des SSMS wesentlich schneller auszurollen, da nicht mehr auf ein Update des großen Bruders, SQL Server, gewartet werden muss.

Auch ist es so möglich, schneller an den Neuerungen, die zunächst in Azure veröffentlicht werden, teilzuhaben.

1.4 SQL Server 2017 – wichtige Neuerungen im Überblick

Mit SQL Server 2017 tat Microsoft einen großen, strategischen Schritt. Zum ersten Mal ist es möglich, SQL Server auch auf einem Nicht-Microsoft-Betriebssystem zu betreiben. Dieses Novum sorgte unter den Nutzern von Datenbanksystemen für große Aufregung. Dabei durfte aber nicht übersehen werden, was es sonst noch alles an Weiterentwicklungen gab.

SQL Server 2017 ist eine Plattform, auf der Sie verschiedene Wahlmöglichkeiten haben. Sie können verschiedene Entwicklungssprachen und Datentypen wählen, lokal oder in der Cloud ausführen und zwischen verschiedenen Betriebssystemen wählen. Die Leistungsfähigkeit von SQL Server wird unter Linux, in Linux-basierten Docker-Containern oder unter Windows bereitgestellt.

1.4.1 SQL Server 2017: Datenbank-Engine (Database Engine)

Im Bereich der Datenbank-Engine gab es viele Verbesserungen. Hier eine Übersicht:

- *CLR-Assemblies* können jetzt in eine Whitelist eingetragen werden. Dadurch kann die CTP-2.0-Funktion `clr strict security` umgangen werden.
- `sp_add_trusted_assembly`, `sp_drop_trusted_assembly` und `sys.trusted_asssemblies` wurden hinzugefügt, um die Positivliste von vertrauenswürdigen Assemblies zu unterstützen.
- Die fortsetzbare Online-Indexneuerstellung (*Resumable Online Index Rebuild*) kann, wenn sie durch einen Fehler, oder auch durch den Benutzer, unterbrochen wurde, dort fortgesetzt werden, wo sie unterbrochen wurde.
- Mit der Option `IDENTITY_CACHE` für `ALTER DATABASE SCOPED CONFIGURATION` können Sie Lücken in den Werten von Identitätsspalten vermeiden, wenn ein Server unerwartet neu startet oder ein Failover zu einem sekundären Server ausführt.
- Eine neue Generation von Verbesserungen bei der Abfrageverarbeitung verwendet Optimierungsstrategien zur Verbesserung der Laufzeitbedingungen der Auslastung Ihrer Anwendung.
- Drei Verbesserungen sind in dieser ersten Version der adaptiven Abfrageverarbeitung (*adaptive query processing*) zu nennen:
  - Neu sind Adaptive Joins im Batchmodus (*Batch Mode Adaptive Joins*), Feedback zur Speicherzuweisung im Batchmodus (*Batch Mode Memory Grant Feedback*)

und überlappende Ausführung *(Interleaved Execution)* für Tabellenwertfunktionen mit mehreren Anweisungen.

  - Die automatische Datenbankoptimierung (*Automatic Database Tuning*) unterstützt dabei, potentielle Abfrageleistungsprobleme zu finden, empfiehlt Lösungen und kann die gefundenen Probleme sogar automatisch beheben.
  - Neue Graph-Datenbankfunktionen (*Graph Database Capabilities*) für das Modellieren von m:n-Beziehungen beinhalten die Syntax `CREATE TABLE` zum Erstellen von Knoten und Edgetabellen und das Schlüsselwort `MATCH` für Abfragen.

- In der Konfiguration, auslesbar mit `sp_configure`, ist die Option **`clr`** `strict security` standardmäßig aktiv. Dadurch wird die Sicherheit von CLR-Assemblies verbessert.
- Die TempDB kann jetzt schon im Setup mit einer Dateigröße von bis zu 256 GB pro Datei angegeben werden. Außerdem gibt das Setup eine Warnung aus, wenn die Dateigröße mit mehr als 1 GB angegeben, IFI (*Initial File Initialization*) aber nicht aktiviert ist.
- In der *Management-View* `sys.dm_db_file_space_usage` gibt es die neue Spalte `modified_extent_page_count`, die differenzielle Änderungen in jeder Datenbankdatei verfolgt und dabei intelligente Sicherungslösungen wie z. B. eine differenzielle oder eine vollständige Sicherung, basierend auf dem Prozentsatz der geänderten Seiten in der Datenbank, erstellen kann.
- Mithilfe des Schlüsselworts `ON` unterstützt die T-SQL-Syntax jetzt das Laden einer Tabelle in eine andere Dateigruppe mit dem Kommando `SELECT INTO`.
- Datenbankübergreifende Transaktionen (*Cross-Database Transactions*) zwischen Datenbanken einer AlwaysOn Availability Group oder auch einer Instanz werden unterstützt.
- Verfügbarkeitsgruppen (*Availability Groups*) funktionieren jetzt auch ohne Cluster und können bei Migrationen zwischen verschiedenen Betriebssystemen (Windows/Linux) unterstützen.
- Diese neuen dynamischen Verwaltungssichten gibt es jetzt:
  - `sys.dm_db_log_stats` fasst Ebenenattribute und Informationen über die Transaktionsprotokolle zusammen. Dies kann hilfreich sein, um die Integrität der Transaktionsprotokolle zu überwachen.
  - `sys.dm_db_log_info` zeigt Ihnen die VLF-Informationen (Virtual Log File).
  - `sys.dm_tran_version_store_space_usage` hilft Ihnen dabei, den Überblick über die Versionsspeichernutzung pro Datenbank zu behalten (Stichwort: Größe der TempDB).

  - `sys.dm_db_stats_histogram` bietet eine dynamische Verwaltungsansicht, damit Sie die Statistiken besser untersuchen können.
  - `sys.dm_os_host_info` zeigt Ihnen die Systeminformationen sowohl für Windows als auch für Linux.

- Der Datenbankoptimierungsratgeber *(Database Tuning Advisor, DTA)* wurde in der Leistung verbessert und hat neue Optionen erhalten.
- In-Memory-Erweiterungen (*In-Memory Enhancements*) umfassen auch die Unterstützung für berechnete Spalten in speicheroptimierten Tabellen, die vollständige Unterstützung für JSON-Funktionen in nativ kompilierten Modulen und den **`CROSS APPLY`**-Operator in nativ kompilierten Modulen.
- Es gibt neue Zeichenfolgenfunktionen (*String Functions*): `CONCAT_WS`, `TRANSLATE` und `TRIM`, und der Filter `WITHIN GROUP` wird jetzt für die Funktion `STRING_AGG` unterstützt.
- `BULK INSERT` und `OPENROWSET(Bulk…)` wurden als Massenzugriffooptionen für CSV- und Azure-BLOB-Dateien eingeführt.
- Sie können ab sofort Speicheroptimierte Objektverbesserungen (*Memory-Optimized Object Enhancements*) auf Azure Storage speichern.
- `DATABASE SCOPED CREDENTIAL` wurden erweitert.
- Passend zum SQL Server 2016 wurde der neue Datenbank `COMPATIBILITY_LEVEL 140` hinzugefügt.

1.4.2 SQL Server 2017: Integration Services (SSIS)

- Die *Scale-Out-Funktion* wurde den Integration Services hinzugefügt. Dabei unterstützt *Scale Out Master* die Hochverfügbarkeit. Die *Scale Out Worker* wurden hinsichtlich des Failovers verbessert.
- Der Parameter `runincluster` der Prozedur `[catalog].[create_execution]` wird in `runinscaleout` umbenannt. So werden Konsistenz und Lesbarkeit verbessert.
- Die *Integration Services* von SQL Server 2017 unterstützen jetzt auch SQL Server unter Linux, und SSIS-Pakete können auf Linux per Befehlszeile ausgeführt werden.
- *Scale Out* für SSIS vereinfacht die Ausführung der Integration Services auf mehreren Servern.
- *OData-Quelle* und der *OData-Verbindungsmanager* unterstützen jetzt Verbindungen mit den *OData-Feeds* von Microsoft Dynamics AX Online und Microsoft Dynamics CRM Online

1.4.3 SQL Server 2017: Master Data Services (MDS)

Die Anwenderfreundlichkeit und Leistung beim Upgrade von SQL Server 2012, 2014 und 2016 auf die *Master Data Services 2017* sind deutlich verbessert.

- Die sortierte Liste der Entitäten, Sammlungen und Hierarchien kann auf der Explorer-Seite angezeigt werden.
- Die Leistung beim Erweitern des Ordners ENTITÄTEN (en.: ENTITIES) auf der Seite GRUPPEN VERWALTEN (en.: GROUP MANAGEMENT) zum Zuweisen von Modellberechtigungen wurde verbessert, und die Seite GRUPPEN VERWALTEN befindet sich im Abschnitt SICHERHEIT (en.: SECURITY) der Webanwendung.

1.4.4 SQL Server 2017: Analysis Services (SSAS)

SQL Server Analysis Services 2017 führt zahlreiche Verbesserungen für tabellarische Modelle ein. Hier eine Liste der Verbesserungen:

- Der tabellarische Modus ist jetzt die Standard-Installationsoption für die Analysis Services.
- Das Absichern der Metadaten von tabellarischen Modellen auf Objektebene wurde optimiert.
- Beziehungen basierend auf Datumsfeldern können einfach mit Datumsabhängigkeiten erstellt werden.
- Neue Datenquellen **`Get Data`** (*Power Query*) und die vorhandene Unterstützung der *DirectQuery*-Datenquellen für M-Abfragen wurden ergänzt.
- DAX-Editor für SSDT ist neu hinzugekommen.
- Die Funktionen der Optimierung der Datenaktualisierungen von großen tabellarischen In-Memory-Modellen mit Codierungshinweisen, wurde erweitert.
- Der *Kompatibilitätsgrad 1400* für tabellarische Modelle wird unterstützt. Zur Erstellung oder für das Upgrade vorhandener Projekte auf den Kompatibilitätsgrad 1400 müssen Sie allerdings die SQL Server Data Tools (SSDT) 17.0 RC2 herunterladen und installieren.
- **`Get Data`** für tabellarische Modelle bieten eine moderne Erfahrung mit dem Kompatibilitätsgrad 1400.
- Leere Elemente in unregelmäßigen Hierarchien können mit der Eigenschaft ELEMENTE-AUSBLENDEN ausgeblendet werden.
- Endbenutzer können Details für aggregierte Informationen in den neuen Detailzeilen anzeigen.
- Der Operator **`DAX IN`** für die Angabe mehrerer Werte.

1.4.5 SQL Server 2017: Reporting Services (SSRS)

- Die *Reporting Services der Version 2017* können nicht mehr über das SQL-Server-Setup installiert werden. Stattdessen können Sie sie als eigenständiges Paket bei Microsoft herunterladen.
- Sie können jetzt Kommentare in Berichte einfügen und damit die Zusammenarbeit mit anderen Benutzern verbessern. Außerdem können Sie Anlagen in die Kommentare einfügen.
- Native DAX-Abfragen für verschiedene Tabellendatenmodelle des *SSAS* können Sie in der neuesten Version des Berichtsgenerators erstellen, indem Sie die gewünschten Felder einfach in den Abfrage-Designer ziehen.
- *SSRS* unterstützt eine vollständige, mit OpenAPI kompatible RESTful-API, um die Entwicklung von modernen Anwendungen und Anpassungen zu ermöglichen.

1.4.6 SQL Server 2017: Machine Learning

Die *SQL Server R Services* wurden in *SQL Server Machine Learning Services* umbenannt. Damit soll auf die zusätzliche *Python*-Unterstützung hingewiesen werden.

- Die Machine Learning Services können im SQL Server Python- oder R-Skripte ausführen. Es ist aber auch möglich, den Machine Learning Server als eigenständiges Tool zu installieren und so die Nutzung von Python- und R-Skripts ermöglichen, wo kein SQL-Server-Service erforderlich ist.
- Entwickler von SQL Server haben jetzt Zugriff auf die umfangreichen ML- und AI-Bibliotheken für Python, die in der Open-Source-Umgebung zusammen mit den neuesten Innovationen von Microsoft zur Verfügung stehen.
- *revoscalepy* ist die Python-Entsprechung zu *RevoScaleR* und enthält parallele Algorithmen für lineare und logistische Regressionen, Entscheidungsstrukturen, verstärkte Strukturen und zufällige Gesamtstrukturen sowie einen umfangreichen Satz an APIs für die Übertragung und Verschiebung von Daten, Remoterechenkontexte und Datenquellen.
- *microsoftml* ist ein modernes Paket von ML-Algorithmen und -Transformationen mit Python-Bindungen. Es enthält tiefgreifende neuronale Netzwerke, schnelle Entscheidungsstrukturen und Entscheidungsgesamtstrukturen sowie optimierte Algorithmen für lineare und logistische Regressionen. Darüber hinaus erhalten Sie vorgegebene, auf ResNet-Modellen basierende Modelle, die Sie zur Imageextraktion oder Standpunktanalyse verwenden können.
- Mithilfe der gespeicherten Prozedur `sp_execute_external_script` können Sie Python-Skripts ganz leicht mit T-SQL bereitstellen. Profitieren Sie von hervorragender Leistung, indem Sie Daten aus SQL an Python-Prozesse streamen und MPI-Ringparallelisierung verwenden.

- Datenanalysten und Entwickler können Python-Code remote aus ihrer Entwicklungsumgebung ausführen, um Daten- und Entwicklungsmodelle auszuprobieren, ohne dabei Daten zu verschieben.
- Native Bewertung: Die **`PREDICT`**-Funktion in Transact-SQL kann in jeder Instanz von SQL Server 2017 zum Durchführen von Bewertungen verwendet werden, auch wenn *R* nicht installiert ist. Dafür müssen Sie nur das Modell mit einem der unterstützten *RevoScaleR*- und *revoscalepy*-Algorithmen trainieren und es in einem neuen, kompakten Binärformat speichern.
- Die Paketverwaltung *T-SQL* unterstützt jetzt die **`CREATE EXTERNAL LIBRARY`**-Anweisung, um Datenbankadministratoren bessere Verwaltungsfunktionen für *R*-Pakete zu bieten. Verwenden Sie Rollen, um den Zugriff auf private oder freigegebene Pakete zu steuern, speichern Sie R-Pakete in der Datenbank, und geben Sie diese für Benutzer frei.
- Die gespeicherte Prozedur `sp_execute_external_script` wurde optimiert und unterstützt jetzt die Batchmodusausführung für Columnstore-Daten.

1.5 SQL Server 2019 – Aussichten auf die Neuerungen im Überblick

SQL Server 2019 bietet eine große Auswahl an Entwicklungssprachen, Datentypen und Betriebssystemen sowie die Möglichkeit, lokal oder in der Cloud zu arbeiten.

SQL Server 2019 bietet folgende neue Funktionen (weitere Informationen zu diesen Features finden Sie weiter unten):

CTP 2.1

- Der Big Data Cluster wurde erweitert.
- Python- und R-Apps werden bereitgestellt.
- Die Datenbank-Engine wurde verbessert.
- Die intelligente Abfrageverarbeitung fügt Inlining für benutzerdefinierte Skalarfunktionen hinzu.
- Die Fehlermeldung beim Abschneiden von Daten mit Tabellen- und Spaltennamen sowie abgeschnittenen Werten wurden verbessert.
- SQL-Server-Setup unterstützt UTF-8-Sortierungen.
- Abgeleitete Tabellen- oder Ansichtsaliasse in Graphabgleichsabfragen können verwendet werden.
- Die Diagnosedaten für gesperrte Statistiken wurden verbessert.
- Ein neuer hybrider Pufferpool wurde eingefügt.
- Die statische Datenmaskierung wurde hinzugefügt.
- Azure Data Studio wurde weiterentwickelt.

CTP 2.0

- Big Data Cluster wurde verbessert.
- Bereitstellen eines Big Data Clusters mit SQL und Spark-Linux-Containern in Kubernetes
- Der Zugriff auf Big Data über HDFS wurde erneuert.
- Das Ausführen von erweiterten Analysen und Machine-Learning-Vorgängen mit Spark ist jetzt möglich.
- Das Streamen von Daten an SQL-Server-Datenpools mit Spark wurde eingeführt.
- Abfragebüchern können zur Bereitstellung einer Notebookumgebung mit Azure Data Studio ausgeführt werden.
- Die Datenbank-Engine wurde erneut verbessert.
- Die Unterstützung für UTF-8 wurde verbessert.
- Das Fortsetzen der Indexerstellung nach einer Unterbrechung durch die Erstellung von fortsetzbaren Onlineindizes wurde ergänzt.
- Die Erstellung und Neuerstellung von gruppierten Columnstore-Onlineindizes wurden überarbeitet.
- Always Encrypted kann jetzt mit Secure Enclaves umgehen.
- Die intelligente Abfrageverarbeitung wurde optimiert.
- Die Programmierbarkeit der Java-Sprache wurde erweitert.
- SQL-Graphfeatures wurden ergänzt.
- Eine datenbankweit gültige Konfigurationseinstellung für Online- und fortsetzbare DDL-Vorgänge wurde ergänzt
- AlwaysOn-Verfügbarkeitsgruppen beherrschen jetzt die Umleitung von Verbindungen mit sekundären Replikaten
- Neu ist die nativ in SQL Server integrierte Datenermittlung und -klassifizierung.
- Die Unterstützung für Geräte mit persistentem Speicher wurde erweitert.
- Eine Unterstützung für Columnstore-Statistiken in **`DBCC CLONEDATABASE`** wurde ergänzt.
- Es gibt jetzt neue Optionen für `sp_estimate_data_compression_savings`.
- Die SQL Server Machine Learning Services funktionieren jetzt im Failover Cluster.
- Neu ist auch die standardmäßig aktivierte *LWP*-Abfrageinfrastruktur (*Lightweight Profiling*).
- Neue PolyBase-Connectors wurden ergänzt.
- Die Rückgabe von Seiteninformationen durch neue `sys.dm_db_page_info`-Systemfunktion wurde hinzugefügt.
- SQL Server kann jetzt auch unter Linux installiert werden.

- Eine erweiterte Replikationsunterstützung wurde ergänzt.
- Die Unterstützung für den *Microsoft Distributed Transaction Coordinator* (*MSDTC*) wurde verbessert.
- AlwaysOn-Verfügbarkeitsgruppe in Docker-Containern mit Kubernetes sind nun möglich.
- OpenLDAP-Unterstützung für AD-Drittanbieter wurde eingeführt.
- Machine Learning ist jetzt auch unter Linux möglich.
- Es gibt eine neue Containerregistrierung.
- Neue RHEL-basierte Containerimages wurden ergänzt.
- Die Benachrichtigung zur Speicherauslastung wurde erweitert.
- Die Master Data Services wurden deutlich verbessert.
- Die Ersetzung der Silverlight-Steuerelemente wurde umgesetzt.
- Security: auch in diesem Bereich hat es Verbesserungen gegeben
- Die Zertifikatverwaltung im SQL-Server-Konfigurations-Manager wurde optimiert.
- *SQL Server Management Studio* (*SSMS*) *18.0* (Vorschau)
- Eine neue Version von Azure Data Studio wurde veröffentlicht.

Big Data Cluster

Der SQL Server 2019 Big Data Cluster ermöglicht folgende neue Szenarien:

- Bereitstellen von Python- und R-Apps (CTP 2.1)
- Bereitstellen eines Big Data Clusters mit SQL-Server- und Spark-Linux-Containern in Kubernetes (CTP 2.0)
- Zugriff auf Big Data über HDFS (CTP 2.0)
- Ausführen von erweiterten Analysen und Vorgängen des maschinellen Lernens mit Spark (CTP 2.0)
- Streamen von Daten an SQL-Datenpools mit Spark (CTP 2.0)
- Ausführen von Abfragebüchern zur Bereitstellung einer Notebookumgebung in Azure Data Studio (CTP 2.0)

Hinweis

Der *SQL Server 2019 Big Data Cluster* ist zunächst als eingeschränkte öffentliche Vorschauversion verfügbar. Wenn sie an Tests mit dem Big Data Cluster interessiert sind, müssen Sie sich unter folgender URL registrieren:

https://sqlservervnexteap.azurewebsites.net/.

Kapitel 7

Grundlegendes Know-how für Administratoren und Entwickler

Bei der Einarbeitung in ein Datenbanksystem gibt es viele Informationen, Features und Begriffe, die anfangs ungefiltert auf einen Administrator oder Entwickler einströmen. Wo also anfangen – was ist wirklich wichtig? Dieses Kapitel erläutert Ihnen, was für die Administration von SQL Server und das Entwickeln von Datenbankanwendungen für dieses System von Bedeutung ist.

SQL Server bietet eine Vielzahl an Tools, Befehlen und Konfigurationsmöglichkeiten. Es stellt sich demnach die Frage, womit man bei der Einarbeitung in dieses Datenbanksystem am besten beginnt, um SQL Server zu administrieren oder um in die Entwicklung von Datenbankanwendungen einzusteigen.

In diesem Kapitel stellen wir deshalb wichtige Begriffe, Fragen und Informationen vor, mit denen Sie als Administrator oder Entwickler von SQL-Server-Lösungen früher oder später konfrontiert werden. Dabei geht es zuerst um das zentrale Verwaltungsprogramm: das *SQL Server Management Studio*. Darüber hinaus werden wir Ihnen aber auch *Trigger*, *gespeicherte Prozeduren* und *Cursorschleifen* näher erläutern. Dieses Wissen ist wichtig und wird Sie dabei unterstützen, nötige Zusammenhänge zu erkennen und noch leistungsfähigere Datenbankapplikationen zu schreiben.

7.1 SQL Server verwalten – SQL Server Management Studio (SSMS)

Ein wichtiges Tool und Werkzeug für Administratoren und Entwickler ist das Verwaltungstool *SQL Server Management Studio* (*SSMS*).

Beim SSMS handelt es sich um das zentrale Verwaltungstool, das dem Administrator die Konfiguration und Steuerung von SQL Server, aber auch das Entwickeln von Datenbanklösungen ermöglicht.

Mit dem SSMS setzen Sie SQL-Server-Konfigurationsparameter, führen Datenbanksicherungen durch, entwickeln Transact-SQL-Skripte (T-SQL) und vieles mehr. Im Folgenden stellen wir Ihnen wesentliche Funktionalitäten des SSMS vor.

7.1.1 Starten des SQL Server Management Studios

Ist SQL Server inklusive der Verwaltungstools auf Ihrem Server installiert, finden Sie das SSMS in der Programmgruppe Start • Alle Programme • Microsoft SQL Server Tools 18 • Microsoft SQL Server Management Studio 18. Wobei 18 die jeweils aktuelle, installierte Versionsnummer ist.

Wenn Sie das Management Studio starten, wird es standardmäßig unter dem User ausgeführt, unter dem Sie am Rechner angemeldet sind. Manchmal ist es aber erforderlich, das Management Studio unter einem anderen Benutzer zu starten, zum Beispiel einem, der Zugriffsrechte auf ihre Instanz hat, oder der über weitergehende Berechtigungen (z. B. SA oder Security Admin) verfügt. In diesem Fall kopieren Sie den Start-Button des Management Studios in die Taskbar und klicken dann mit der linken Maustaste bei gedrückter ⇧-Taste auf den Button. In der Auswahl, die jetzt erscheint, klicken Sie auf als anderer Benutzer starten (englisch: Run as different User).

7.1.2 Grundlegender Aufbau des SQL Server Management Studios

Nachdem Sie das SSMS gestartet und sich mit Ihren Anmeldedaten angemeldet haben, befinden Sie sich in der SSMS-Oberfläche, die im Wesentlichen aus vier Bereichen aufgebaut ist (siehe Abbildung 7.1). Im oberen Bereich befinden sich wie üblich Menüs und Symbolleisten ❶.

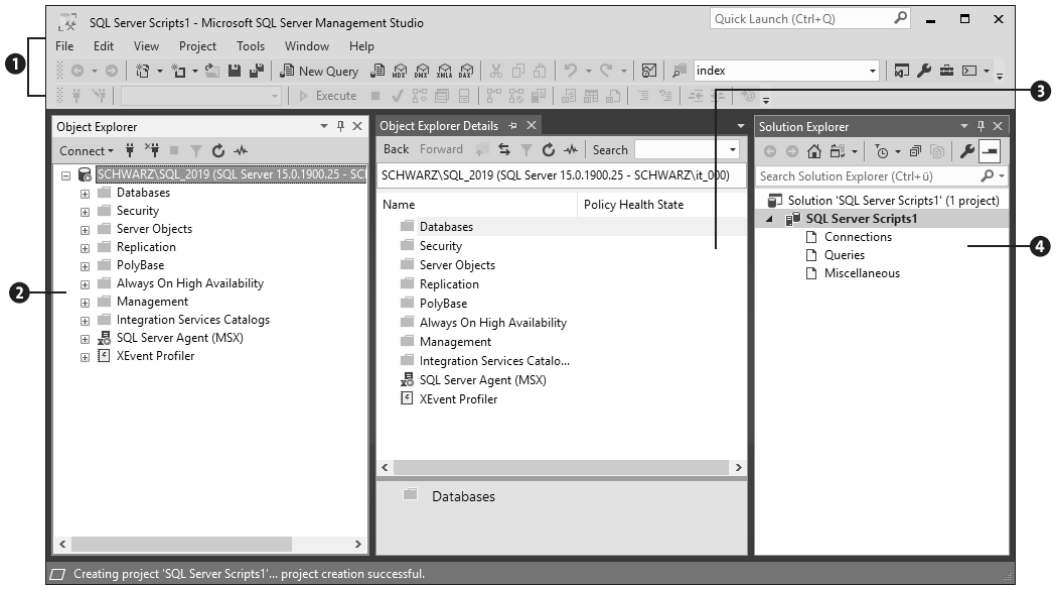

Abbildung 7.1 Aufbau des SQL Server Management Studios

Ein Datenbankserver verwaltet in erster Linie Daten. Diese Daten sind in Tabellen und die Tabellen wiederum in den Datenbanken organisiert. Daneben gibt es eine ganze Palette an weiteren und unterstützenden Datenbankobjekten. Diese Server- und Datenbankobjekte in ihrer logischen Anordnung darzustellen ist Aufgabe des *Objekt-Explorers* ❷, der sich im linken Bereich der SSMS-Oberfläche befindet. Der Objekt-Explorer stellt die Objekte in einer hierarchischen Baumstruktur dar und ermöglicht somit ein schnelles Navigieren innerhalb der SQL-Server-Struktur.

Der mittlere Bereich ❸ bildet den eigentlichen Arbeitsbereich. Im sogenannten *Abfrage-Editor-Fenster* (*Query Editor*) geben Sie die eigentlichen T-SQL-Befehle ein, oder es werden je nach Auswahl der gewünschten Funktion entsprechende Dialoge und Informationen angezeigt, die eine Konfiguration von SQL Server ermöglichen.

Im rechten Bereich ❹ können Sie weitere Infofenster und Dialoge einblenden, die Sie bei der Anzeige und Arbeit unterstützen. Unterschiedliche Hilfefunktionalitäten, Eigenschaftsfenster oder Projektarbeitsmappen gehören dazu.

Die Anzeige der soeben beschrieben Eigenschaftsfenster, Symbolleisten etc. können Sie über die Menüpunkte im Menü Anzeige (View) des SSMS anpassen. Darum ist Ihr SSMS möglicherweise anders aufgebaut als in Abbildung 7.1.

7.1.3 Der Objekt-Explorer

Der Objekt-Explorer im linken Bereich der SSMS-Oberfläche stellt alle Server- und Datenbankobjekte in einer übersichtlichen hierarchischen Baumstruktur dar, die Sie in Abbildung 7.2 beispielhaft sehen.

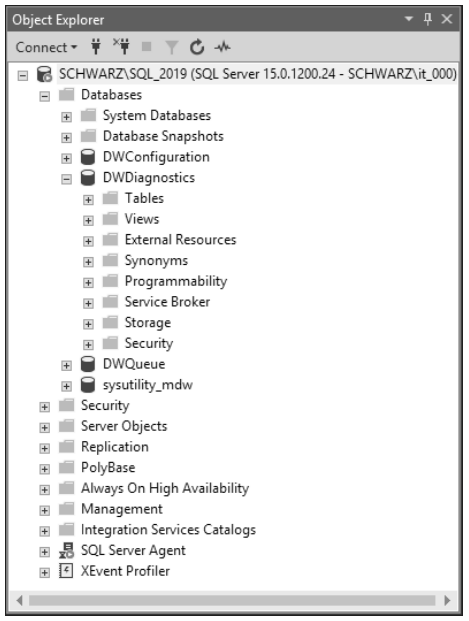

Abbildung 7.2 Der Objekt-Explorer im SQL Server Management Studio

Durch einen Klick auf die einzelnen Knoten navigieren Sie innerhalb der Baumstruktur und wählen die einzelnen Datenbankobjekte und weitere Funktionalitäten aus. Der oberste Ordner (in Abbildung 7.2 der Ordner SCHWARZ\SQL_2019) stellt dabei immer die Ebene der aktuellen Serverinstanz dar. Öffnen Sie diesen Ordner, sehen Sie weitere Ordner, die Ihnen die Verwaltung der Serverinstanz und deren Datenbanken ermöglichen. Klicken Sie wiederum auf einen Datenbankordner – in unserem Beispiel aus Abbildung 7.2 ist das der Ordner DWDIAGNOSTICS –, werden Ihnen weitere Unterordner angezeigt, wie z. B. die Ordner TABELLEN (TABLES), SICHTEN (VIEWS) etc., die auf der Ebene der jeweiligen Datenbank (hier DWDIAGNOSTICS) relevant sind.

Viele Eigenschaften von Datenbankobjekten, wie z. B. die Eigenschaften von Tabellen, setzen Sie, indem Sie mit der rechten Maustaste auf ein Datenbankobjekt klicken. Das daraufhin erscheinende Kontextmenü zeigt Ihnen die Menüpunkte entsprechend den zur Verfügung stehenden Möglichkeiten des gewählten Datenbankobjektes an. Zum Beispiel können Sie sich durch einen Rechtsklick auf eine Datenbanktabelle und durch Auswahl des Menü-Items OBERSTE 1000 ZEILEN AUSWÄHLEN (SELECT TOP 1000 ROWS) die Inhalte der Tabelle anzeigen lassen oder fügen durch Auswahl von ENTWERFEN (DESIGN) neue Spalten zur Tabelle hinzu (siehe Abbildung 7.3).

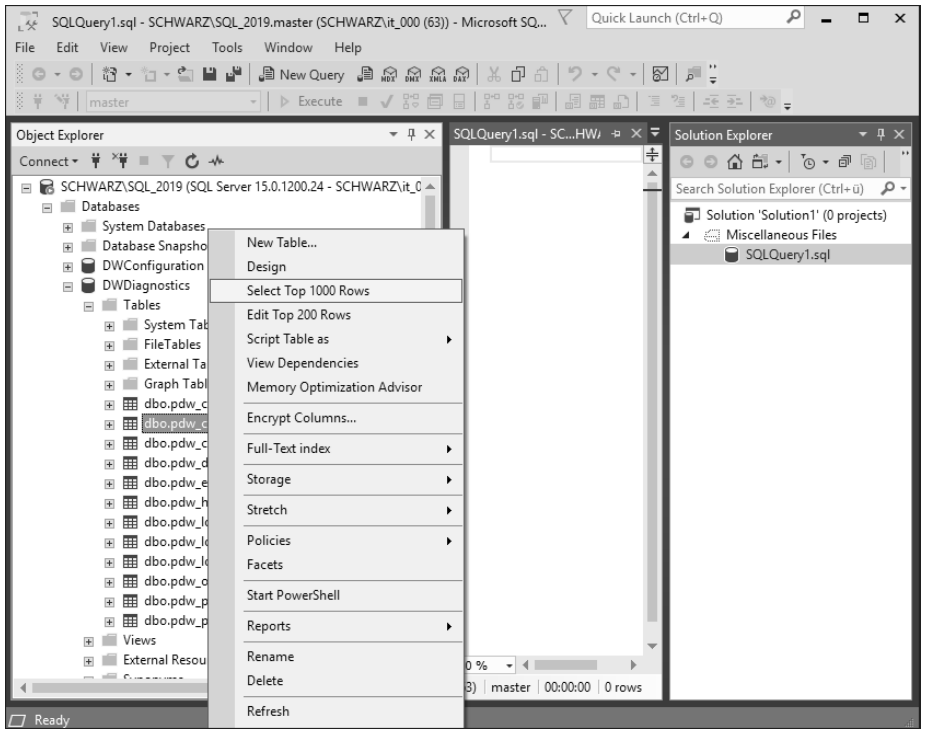

Abbildung 7.3 Kontextmenü des Objekt-Explorers für die Tabelle »pdw_component_health_data«

7.1.4 Abfrage-Editor-Fenster

Wie bereits in Abschnitt 7.1.2, »Grundlegender Aufbau des SQL Server Management Studios«, vorgestellt, befindet sich in der Mitte des SSMS der Arbeitsbereich, in dem Abfrage-Editor-Fenster angezeigt werden. In ein Abfrage-Editor-Fenster geben Sie T-SQL-Kommandos ein und können entsprechend den Möglichkeiten von T-SQL komplexe Datenbankskripte erstellen.

Wie Abbildung 7.4 zeigt, können Sie mehrere Abfrage-Editor-Fenster öffnen und diese dann durch einen Klick auf die Registerkarten auswählen.

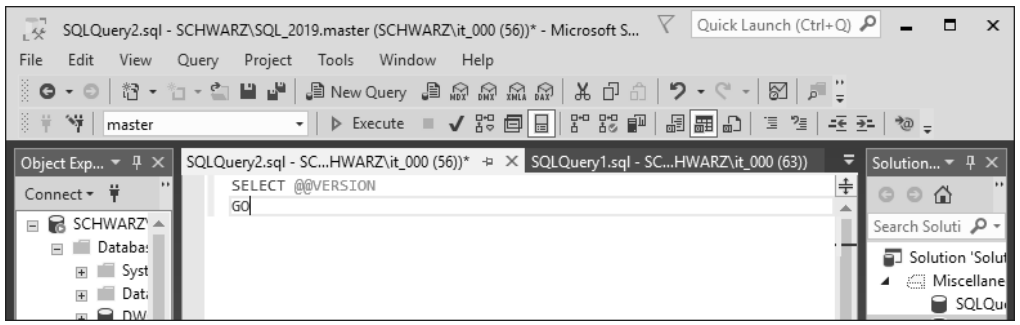

Abbildung 7.4 Abfragefenster und ihre Anordnung im SQL Server Management Studio

Jedes Abfrage-Editor-Fenster kann unter einer anderen SQL-Server-Anmeldung mit SQL Server verbunden sein. Die jeweiligen Verbindungsdaten eines Abfragefensters sehen Sie im mittleren unteren Bereich des SSMS-Fensters (siehe Abbildung 7.5).

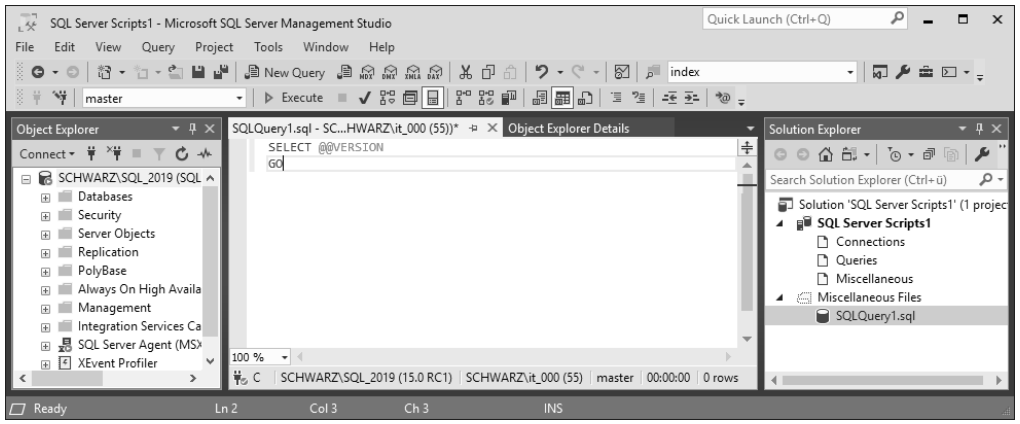

Abbildung 7.5 Verbindungsdatenanzeige im SQL Server Management Studio

Ein neues Abfragefenster erstellen Sie, indem Sie in der Toolbar etwas links von der Mitte auf die Schaltfläche NEUE ABFRAGE (NEW QUERY) klicken (siehe Abbildung 7.6).

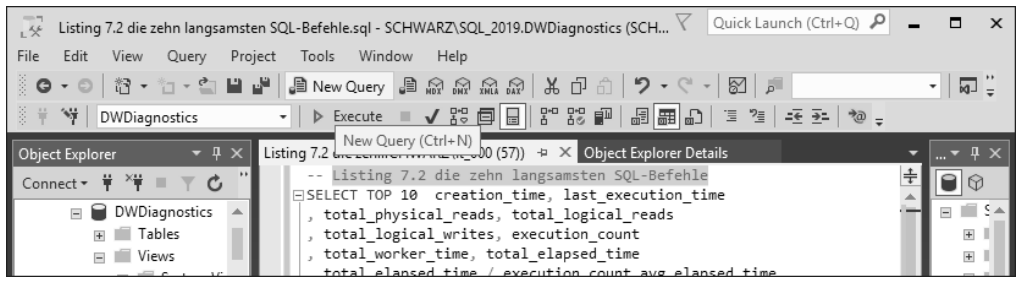

Abbildung 7.6 Neue Abfrage

7.1.5 Management-Studio-Berichte

Ein sehr nützliches und zugleich etwas verstecktes Feature sind die SSMS-Berichte (Reports). Berichte geben Ihnen die Möglichkeit, wichtige SQL-Server-Konfigurations- und Betriebsdaten in schneller und übersichtlicher Form darzustellen. Das Management Studio bietet eine große Anzahl an vorgefertigten Berichten an. Einen solchen Bericht sehen Sie beispielhaft in Abbildung 7.7.

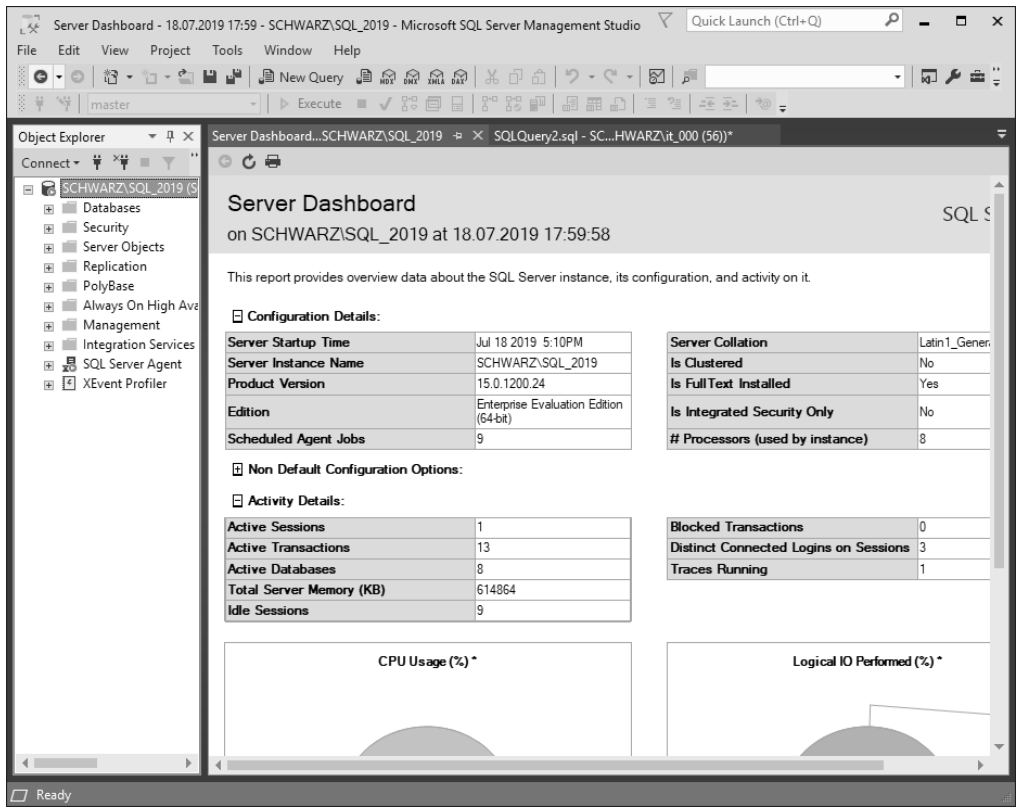

Abbildung 7.7 Bericht »Server Dashboard« im SQL Server Management Studio

Um Berichte aufzurufen, die Ihre Serverkonfiguration anzeigen, gehen Sie wie folgt vor:

1. Klicken Sie im geöffneten Management Studio mit der rechten Maustaste auf Ihren Serverknoten.
2. Im sich öffnenden Kontextmenü wählen Sie den Punkt Berichte (Reports) • Standardberichte (Standard Reports). Sie sehen im daraufhin dargestellten Menü alle vorgefertigten Standardberichte, die Ihre Serverkonfiguration betreffen.
3. Wählen Sie den Bericht Server dashboard aus, der Ihnen wichtige Daten der Grundkonfiguration Ihres Servers anzeigt, wie z. B. CPU-Verwendung und logische E/A.

Was ist unter logischer E/A bzw. I/O zu verstehen?

E/A steht für *Eingabe-Ausgabe*-Operationen. Unter logischen E/A-Vorgängen versteht man das Schreiben und Lesen der Datenseiten aus dem Pufferspeicher (RAM) von SQL Server. Im Englischen spricht man von *Input/Output*-Vorgängen. Die Abkürzung ist in diesem Fall *I/O*, bezeichnet aber den gleichen Vorgang.

Weitere interessante Berichte, wie z. B. den Bericht Datenträgerverwendung (Disc Usage), finden Sie auf der Ebene der Datenbanken.

1. Klicken Sie im geöffneten Management Studio mit der rechten Maustaste auf Ihre Datenbank, deren Datenbankberichte Sie anzeigen möchten.
2. Im sich öffnenden Kontextmenü wählen Sie den Menüpunkt Berichte (Reports) • Standardberichte (Standard Reports). Sie sehen im daraufhin dargestellten Menü alle vorgefertigten Standardberichte, die Ihre Datenbank betreffen.
3. Wählen Sie den Bericht Datenträgerverwendung (Disc Usage), der Ihnen einen Überblick über die Nutzung des Datenbankspeicherplatzes gibt.

Berichte finden Sie im SSMS an den unterschiedlichsten Stellen auf den verschiedenen Ebenen.

7.1.6 Verwalten verschiedener SQL-Serverinstanzen und -versionen

Mit dem SSMS können Sie nicht nur eine Serverinstanz verwalten, sondern durchaus auch mehrere, indem Sie weitere Instanzen zum Objekt-Explorer hinzufügen.

Um innerhalb des SSMS im Objekt-Explorer eine weitere Serverinstanz hinzuzufügen, gehen Sie wie folgt vor:

1. Klicken Sie in der Toolbar des Objekt-Explorers auf die Schaltfläche Verbinden • Datenbankmodul (Connect • Database Engine).

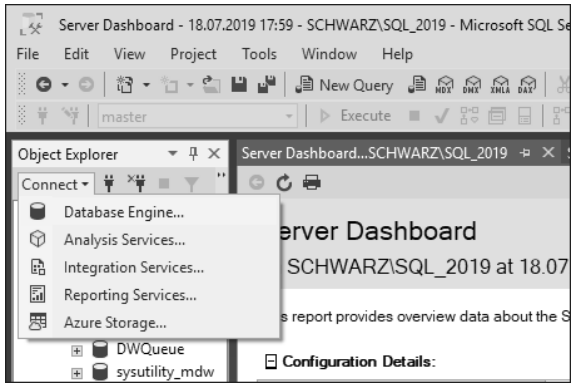

Abbildung 7.8 Objekt-Explorer – Menü »Verbinden« (»Connect«)

2. Im daraufhin erscheinenden Anmeldedialog wählen Sie die gewünschte hinzuzufügende SQL-Server-Instanz inklusive der Anmeldedaten aus und klicken auf Verbinden (Connect).

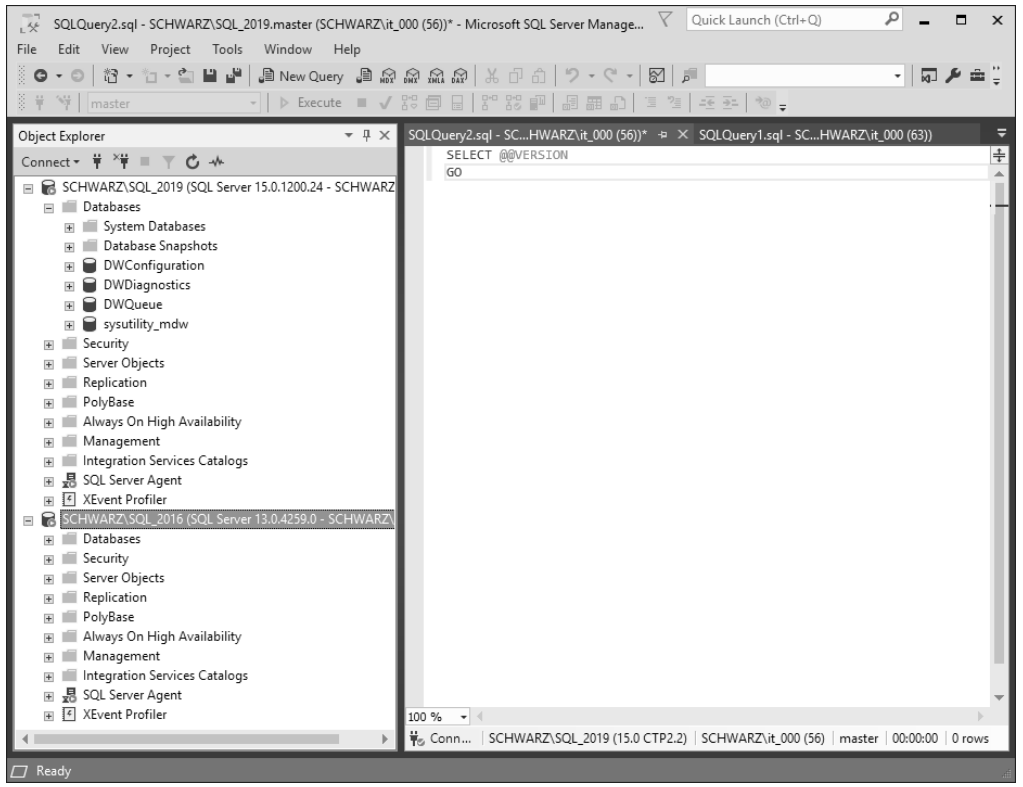

Abbildung 7.9 Weitere zum Objekt-Explorer hinzugefügte Instanz

3. Im Objekt-Explorer wird jetzt ein weiterer Serverinstanzknoten angezeigt (siehe Abbildung 7.9).

Auf diese Art und Weise können Sie auch Serverinstanzen anderer Microsoft-SQL-Server-Versionen, wie z. B. SQL Server 2012 und 2016, hinzufügen und verwalten.

7.1.7 Vorlagen-Explorer und Vorlagenparameter – Schablonen erleichtern die Arbeit

Wenn Sie mit dem SSMS arbeiten, können Sie SQL Server sehr komfortabel über die grafische Benutzeroberfläche verwalten. Dennoch gibt es Aufgaben und Anforderungen, die sich nur schlecht über die durch das SSMS zur Verfügung gestellte Oberfläche erledigen lassen oder vom Management Studio einfach nicht berücksichtigt wurden.

Die Erstellung von Datenbank-Snapshots ist hier ein Beispiel, da es seitens des SSMS keinerlei Unterstützung für das Anlegen von Datenbank-Snapshots gibt. Genaueres zu Datenbank-Snapshots können Sie Abschnitt 9.5, »Datenbank-Snapshots: Datenbankzustände konservieren und wiederherstellen«, entnehmen.

Bei der Erstellung von T-SQL-Datenbankskripten sollte der T-SQL-Code schnell und zuverlässig geschrieben werden können. Für solche Fälle unterstützt Sie das SSMS mit dem Vorlagen-Explorer, den Sie über das Menü Anzeigen • Vorlagen-Explorer (View • Template Explorer) oder über Strg+Alt+T aufrufen.

Der Vorlagen-Explorer hält eine Reihe an vorgefertigten Skripten bereit, die Sie leicht an die eigenen Bedürfnisse anpassen können.

Nach dem Aufruf des SSMS wird der Vorlagen-Explorer im rechten Bereich mit einer Baumstruktur eingeblendet. Die unterschiedlichen Skriptvorlagen sind abhängig vom Typ des Skriptcodes in die Baumstruktur des Vorlagen-Explorers eingegliedert.

Durch Öffnen der einzelnen Ordner können Sie entsprechende Skriptvorlagen auswählen und per Drag & Drop oder durch einen Doppelklick in das Abfragefenster übernehmen. Daraufhin wird im SSMS-Abfrageeditor ein Skript abhängig von der im Vorlagen-Explorer gewählten Skriptvorlage generiert.

In Abbildung 7.10 sehen Sie ein mittels Vorlagen-Explorer erstelltes Skript zum Anlegen einer Datenbank.

Es wurde die Skriptvorlage Create Database (Datenbank erstellen) ausgewählt. Dieses erzeugte Skript enthält den kompletten T-SQL-Befehl der gewählten Aktion.

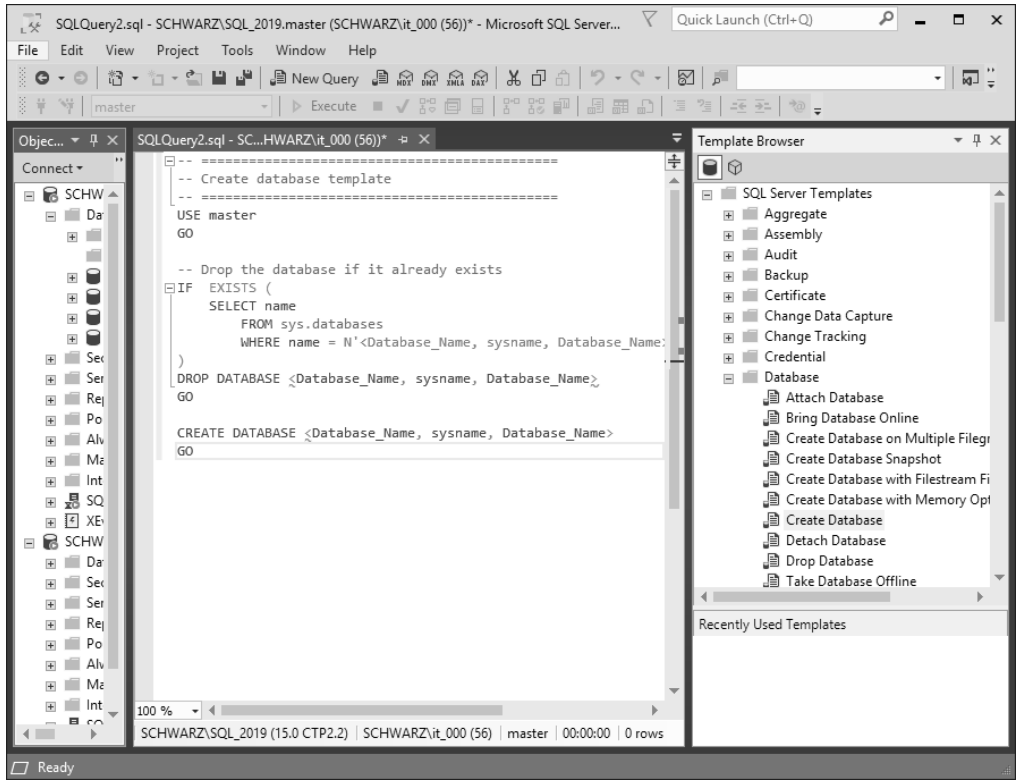

Abbildung 7.10 Vorlagen-Explorer mit per Drag & Drop hinzugefügter Skriptvorlage im Abfragefenster

Vorlagenparameter

Da Sie für die meisten T-SQL-Befehle in der Regel zusätzliche Parameter setzen müssen (das Anlegen einer Datenbank erfordert mindestens die Eingabe des Datenbanknamens), enthält der durch den Vorlagen-Explorer erzeugte Skriptcode markierte Parameterplatzhalter. Diese erkennen Sie an den spitzen Klammern innerhalb des Skriptcodes (siehe Listing 7.1).

```
CREATE DATABASE <Database_Name, sysname, Database_Name>;
```

Listing 7.1 Durch den Vorlagen-Explorer generiertes Skript mit Platzhalter für Parameter

Das SSMS unterstützt Sie auch beim Setzen dieser Parameter. Hierzu wählen Sie im Menü Abfrage (Query) den Punkt Werte für Vorlagenparameter angeben (Specify Values for Template Parameters) aus.

Das SSMS analysiert in diesem Fall den im Abfrageeditor gefundenen T-SQL-Code und zeigt eine abhängig von den gefundenen Parametern gestaltete Dialogbox an. In

diese Dialogbox können Sie die für das Skript benötigten Parameterwerte eingeben (siehe Abbildung 7.11).

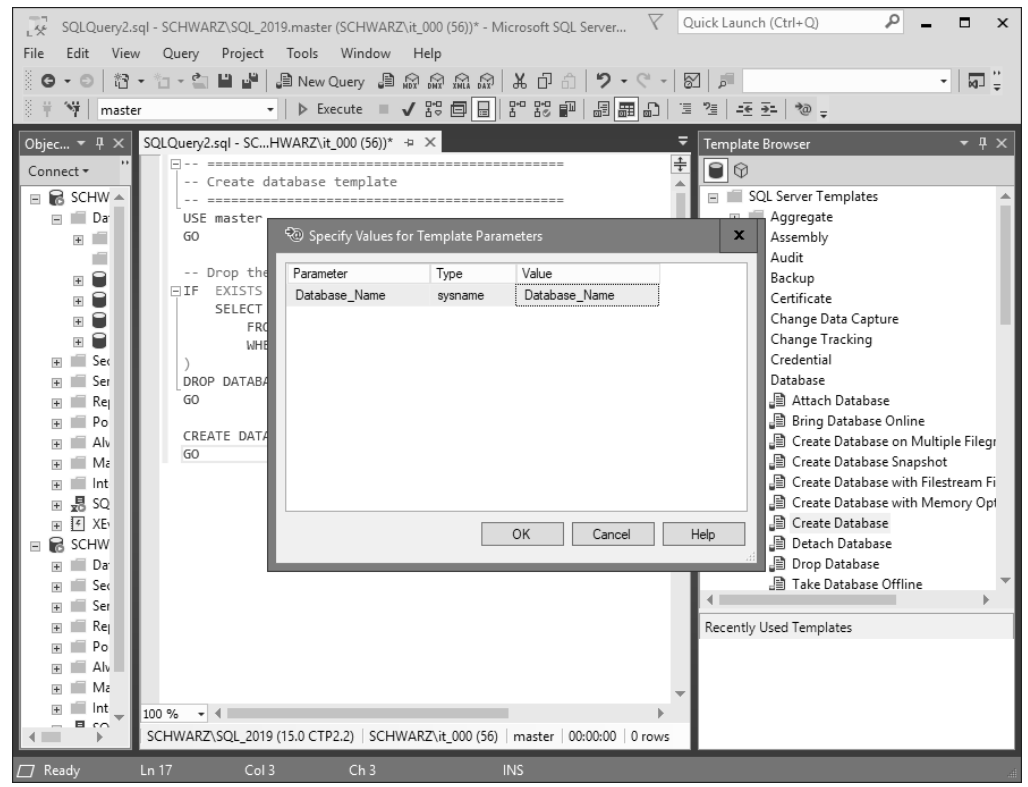

Abbildung 7.11 Der Dialog »Werte für Vorlagenparameter angeben« (»Specify Values for Template Parameters«)

Nach dem Ausfüllen der Spalte Wert (Value) und Anklicken von OK übernimmt das Skript die Parameter.

7.1.8 Der »Skript«-Button – eigentlich ist alles T-SQL

Fast alle Funktionen, die durch die grafische Oberfläche des SSMS zur Verfügung gestellt werden und die die SQL-Server-Instanz oder Datenbankobjekte der Instanz betreffen, werden im Hintergrund direkt in T-SQL-Befehle übersetzt und an den Server gesendet. Wie schön wäre es, wenn es möglich wäre, sich diese T-SQL-Befehle ausgeben zu lassen, um sie in eigenen Skripten zu nutzen. Genau dies ermöglicht das SSMS.

Im SSMS finden Sie in sehr vielen Dialogen im oberen Dialogbereich die Schaltfläche Skript (Script), die Sie auch in Abbildung 7.12 sehen. Mit dieser Schaltfläche lassen Sie sich die vorgenommenen Einstellungen innerhalb des jeweiligen Dialogs in Form des zugehörigen T-SQL-Befehls ausgeben.

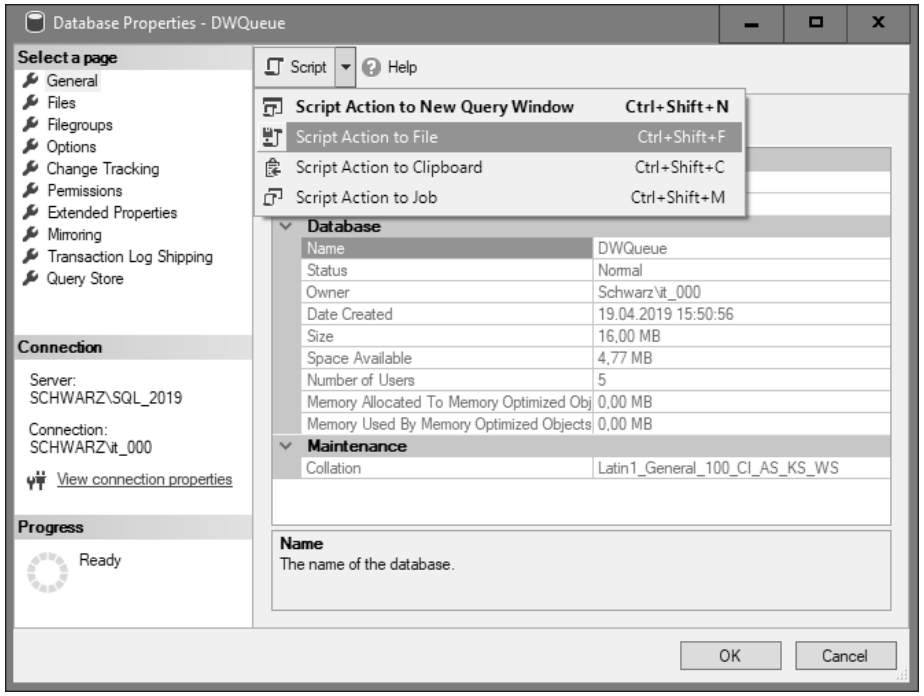

Abbildung 7.12 Schaltfläche »Skript« im Dialogfenster »Datenbankeigenschaften« (»Database Properties«)

Die Schaltfläche Skript (Script) bietet Ihnen mehrere Möglichkeiten zur Auswahl, das über den jeweils aktiven Dialog generierte Skript auszugeben. Zum Beispiel können Sie wählen, ob der generierte T-SQL-Befehl direkt in ein neues Abfragefenster oder in die Zwischenablage geschrieben werden soll. Andere Möglichkeiten wären Skript für Aktion in Datei schreiben (Script Action to File) oder das direkte Anlegen eines Agent-Auftrags, um das generierte Skript automatisiert auszuführen.

> **Hinweis zur Nutzung der Scripting-Funktionalität**
>
> Wichtig ist, dass Sie, wenn Sie das Skript generieren möchten, **nicht** auf die Schaltfläche OK des Dialogs klicken, da sonst die Einstellungen ausgeführt werden und Sie nicht mehr die Möglichkeit haben, die Skript-Schaltfläche (Script) auszuwählen, da der Dialog bei einem Klick auf OK geschlossen und ausgeführt wurde. Der Dialog dient in diesem Fall lediglich dazu, die entsprechenden Skriptanforderungen zu definieren.

Viele Datenbankentwickler oder Administratoren nutzen diese Möglichkeit, um sich die so generierten Befehle ohne langes Studium der Dokumentation ausgeben zu lassen und gegebenenfalls an die eigenen Anforderungen anzupassen.

Ein Skript können Sie auch für ein erstelltes Datenbankobjekt wie eine Tabelle generieren. Hierzu klicken Sie mit der rechten Maustaste auf die Tabelle und wählen aus dem Kontextmenü den Menüpunkt Skript für Tabelle als (Script Table as) aus. Es werden Ihnen unterschiedliche Varianten der Skripterstellung für die Tabelle angezeigt. Sie können wählen, ob Sie ein Skript erstellen möchten, das die Tabelle anlegt, oder eines, das eine Abfrage der Tabelle erlaubt (siehe Abbildung 7.13).

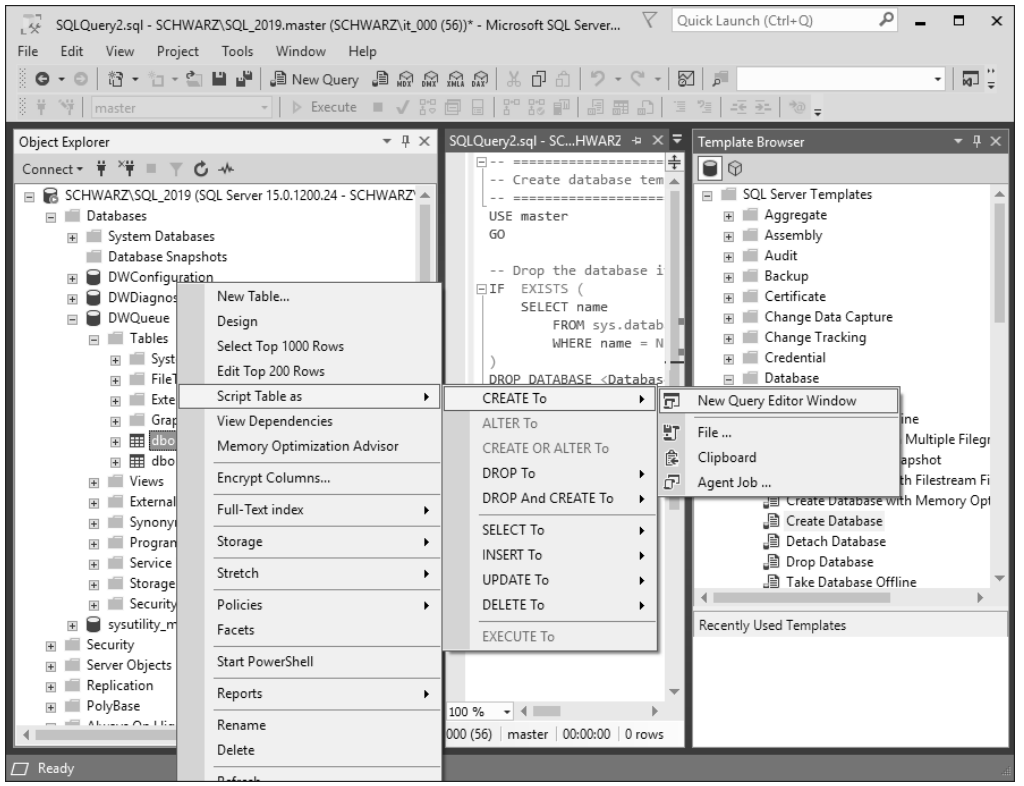

Abbildung 7.13 Skripterstellung für die Tabelle »MessageQueue«

7.1.9 IntelliSense – Unterstützung zur passenden Zeit

Das SSMS bietet IntelliSense-Unterstützung. Unter *IntelliSense* versteht man die automatische Vervollständigung beim Schreiben von Quellcode, die den Programmierer unterstützt.

Die IntelliSense-Funktion des SSMS unterbreitet Ihnen beim Schreiben von Skriptcode Vorschläge zu Datenbankobjekten oder Befehlen. Diese können Sie bei richtigem Vorschlag durch das SSMS direkt mit der ↵- oder der ⇆-Taste übernehmen.

Sie erhalten abhängig von dem von Ihnen eingegebenen Skriptcode unterschiedliche Vorschläge, wie z. B. Namen von Datenbankobjekten oder T-SQL-Befehle. Dies sehen Sie in Abbildung 7.14.

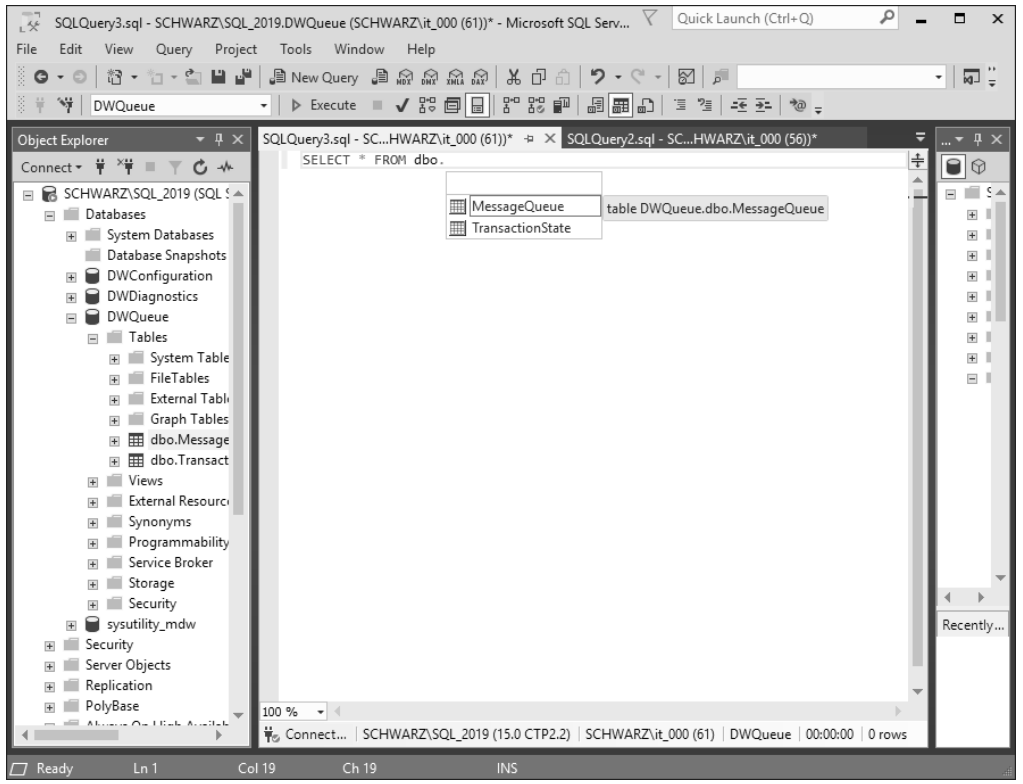

Abbildung 7.14 IntelliSense in Aktion

IntelliSense beschleunigt das Schreiben von Skriptcode und verhindert Tippfehler.

Aktualisieren des IntelliSense-Cache

Sollten Sie einmal einen IntelliSense-Vorschlag z. B. für eine soeben erstellte Tabelle vermissen, können Sie dem lokalen Cache auch manuell durch Auswahl des Menüpunkts Bearbeiten • IntelliSense • Lokalen Cache aktualisieren (Edit • Intellisense • Refresh Local Cache) auf die Sprünge helfen. Dieser wird nämlich nicht immer sofort, sondern in regelmäßigen Abständen aktualisiert.

IntelliSense können Sie im Optionsmenü Extras • Optionen • Text-Editor • Transact-SQL • IntelliSense (Tools • Options • Text Editor • Transact SQL • Intellisense) des SSMS aktivieren und deaktivieren, wie Abbildung 7.15 zeigt.

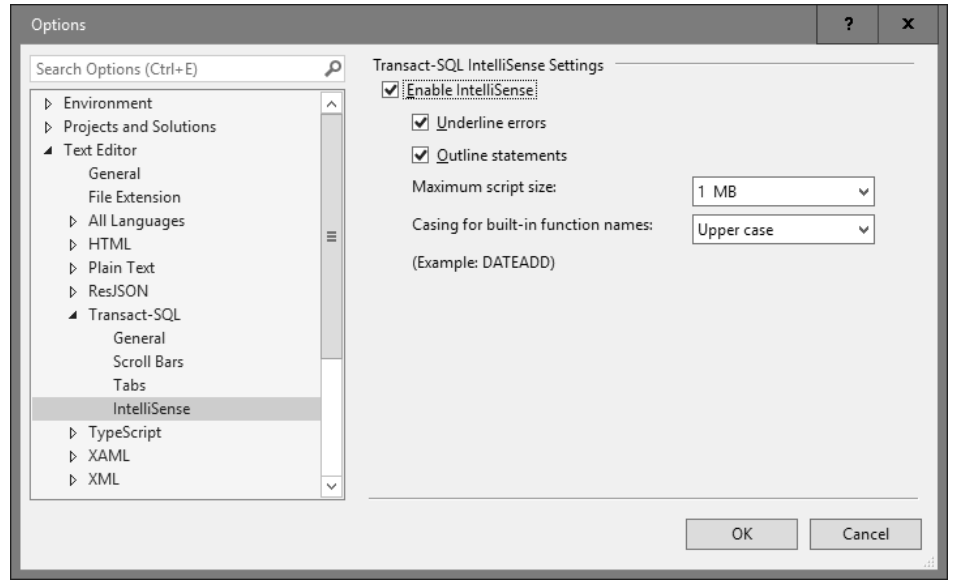

Abbildung 7.15 Einstellungen für IntelliSense im Optionsmenü des SQL Server Management Studios

7.2 Dynamische Verwaltungssichten (Dynamic Management Views, DMV), Katalogsichten

Metainformationen über das Serversystem oder den Datenbankkatalog liefert SQL Server Ihnen in Form von unterschiedlichen Sammlungen von Systemsichten an. Diese Sichten ermöglichen die Abfrage von Metadaten des Datenbanksystems aus unterschiedlichen Perspektiven.

Was sind Metadaten?

Als Metadaten werden Daten bezeichnet, die Informationen über andere Daten enthalten.

7.2.1 Katalogsichten (Catalog Views)

Katalogsichten zeigen Ihnen verschiedene Metadaten über den Datenbankkatalog an. Katalogsichten sind immer dem `sys`-Schema der Datenbank zugeordnet und können nur von den Benutzern abgefragt werden, die die entsprechenden Rechte besitzen. Der Systemadministrator *sa* sieht alle Objekte auf dem Server, und der *dbo* (*database owner*) sieht alle Objekte der Datenbank.

Die Katalogsichten finden Sie im Management Studio im Unterordner Systemsichten (System Views) der jeweiligen Datenbank. Inhalte von Systemsichten lassen Sie sich, wie im SSMS üblich, entweder mit dem SQL-Befehl `SELECT` oder durch einen Rechtsklick auf die Sicht und Auswahl des Menüpunkts Oberste 1000 Zeilen auswählen (Select Top 1000 Rows) anzeigen (siehe Abbildung 7.16).

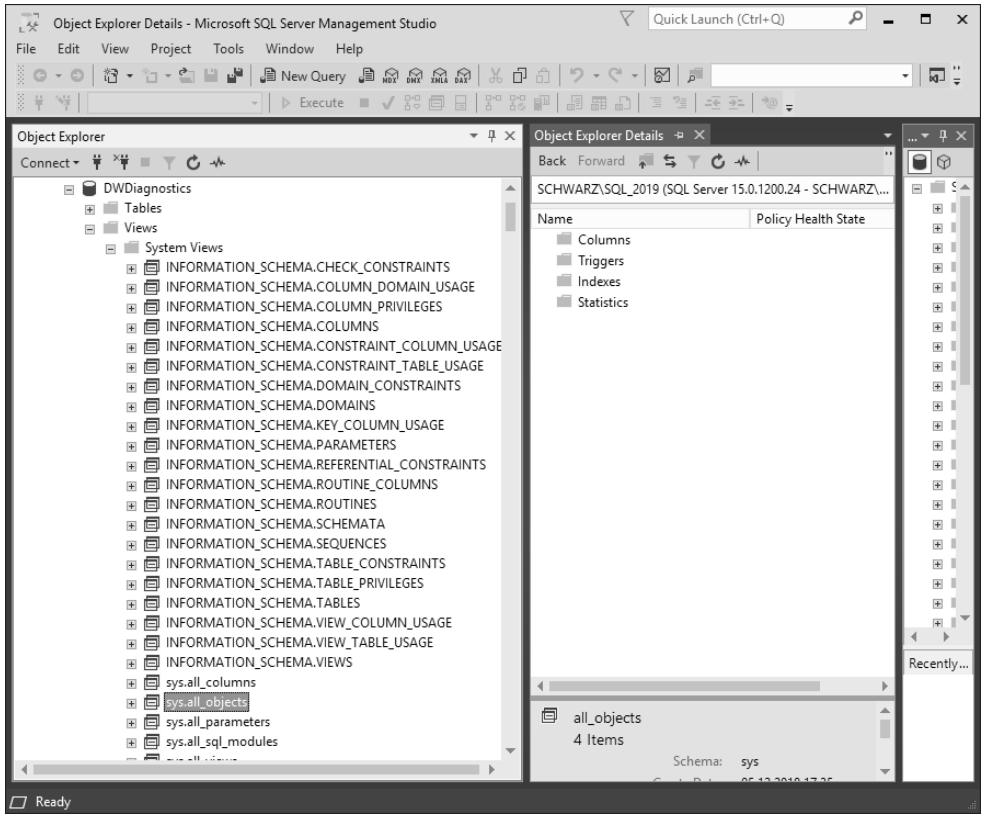

Abbildung 7.16 Katalogsichten-Darstellung im Objekt-Explorer

Um sich z. B. alle Tabellen der aktuellen Datenbank auflisten zu lassen, fragen Sie mit dem Befehl `SELECT * FROM sys.tables` die Katalogsicht `sys.tables` ab.

7.2.2 Dynamische Verwaltungssichten

Dynamische Verwaltungssichten geben Serverstatusinformationen zurück, mit denen Sie in der Lage sind, den Zustand des Servers zu überwachen oder eventuelle Probleme zu diagnostizieren.

Auch diese Sichten sind dem `sys`-Schema zugeordnet, beginnen allerdings mit den Buchstaben `dm_` (steht für *dynamic management*). `SELECT * FROM sys.dm_os_sys_memory` liefert Ihnen z. B. Arbeitsspeicherinformationen des Betriebssystems.

7.2.3 Informationen und Leistungsdaten rundherum

Wenn Sie sich die Systemsichten einmal in Ruhe ansehen, werden Sie feststellen, dass die Informationen innerhalb dieser Sichten sehr umfangreich sind. In diesem Zusammenhang möchten wir darauf hinweisen, dass sämtliche Verwaltungs- und Katalogsichten ausführlich in der SQL-Server-Dokumentation beschrieben sind.

Die aus den Sichten zu gewinnenden Metadaten können Sie sehr gut dazu verwenden, wichtige Informationen zu den Leistungsdaten von SQL Server zu erhalten. Der Bereich der über Systemsichten bereitgestellten Metadateninformationen reicht von Daten über die aktuelle Auslastung des Betriebssystems bis hin zu Informationen über Abfragen, die am meisten Abfragezeit benötigt haben. Diese Informationen erhalten Sie allerdings nur durch eine Kombination oder besser gesagt durch das Bilden von Verknüpfungen (englisch: *joins*) zwischen den einzelnen Systemsichten.

Das Beispielskript in Listing 7.2 verknüpft unterschiedliche dynamische Verwaltungssichten in einer `SELECT`-Abfrage und wertet sie aus. Diese Abfrage ermittelt für Sie die zehn langsamsten SQL-Befehle.

```
SELECT TOP 10 creation_time, last_execution_time
, total_physical_reads, total_logical_reads
, total_logical_writes, execution_count
, total_worker_time, total_elapsed_time
, total_elapsed_time / execution_count avg_elapsed_time
,SUBSTRING(ST.text, (QS.statement_start_offset/2) + 1,
 ((CASE statement_end_offset
 WHEN -1 THEN DATALENGTH(ST.text)
 ELSE QS.statement_end_offset END
 - QS.statement_start_offset)/2) + 1) AS statement_text
FROM SYS.dm_exec_query_stats AS QS
CROSS APPLY SYS.dm_exec_sql_text(QS.sql_handle) AS ST
ORDER BY total_elapsed_time / execution_count DESC;
```

Listing 7.2 Ermitteln der zehn langsamsten SQL-Befehle durch Auswertung dynamischer Verwaltungssichten

7.2.4 Scripting mit T-SQL

In vielen Beispielen innerhalb dieses Buches ist die Rede von *T-SQL*. T-SQL steht für »Transact SQL« und ist Microsofts hauseigene, sehr mächtige Weiterentwicklung des SQL-Standardbefehlssatzes. T-SQL gibt dem Anwender die Möglichkeit, Schleifen, Variablen, Entscheidungsstrukturen und Fehlerbehandlungen innerhalb von T-SQL-Skripten zu definieren.

Was ist ein Skript?

Unter einem Skript versteht man die Aneinanderreihung interpretierbarer Befehle.

Dies ist notwendig, da der normale Befehlssatz des sogenannten *Standard-SQL* die genannten Erweiterungen nicht kennt.

Aus diesem Grund implementieren die unterschiedlichen Hersteller von Datenbanksystemen die ihrer Meinung nach wichtigen Ergänzungsbefehle auf ihre jeweils eigene Art und Weise. So entstehen die sogenannten *SQL-Dialekte*. Viele Datenbanksysteme sprechen ihren eigenen SQL-Dialekt. Während der SQL-Dialekt von Microsoft T-SQL heißt, nennt z. B. Oracle sein erweitertes Standard-SQL PL/SQL.

Wenn Sie im Management Studio daher eine Standard-SQL-Anweisung eingeben, geben Sie eigentlich auch gleichzeitig einen T-SQL-Befehl ein. Die T-SQL-Befehle können Sie einzeln ausführen oder in Form einer Skriptdatei mit der Dateierweiterung *.sql* speichern, um diese Datei dann wieder zu laden und auszuführen. Sobald Sie mehrere T-SQL-Befehle innerhalb einer Datei speichern, schreiben oder ausführen, spricht man von einem *T-SQL-Skript*.

T-SQL-Skripte können in SQL Server vielseitig eingesetzt werden. Sie sind in Funktionen, Triggern und gespeicherten Prozeduren vorhanden. T-SQL-Skripte können auch automatisch über den Agent-Dienst ausgeführt werden.

Möchten Sie ein neues Skript oder Skriptfenster im SSMS öffnen, klicken Sie oben links in der Toolbar auf die Schaltfläche Neue Abfrage (New Query), wie in Abbildung 7.17.

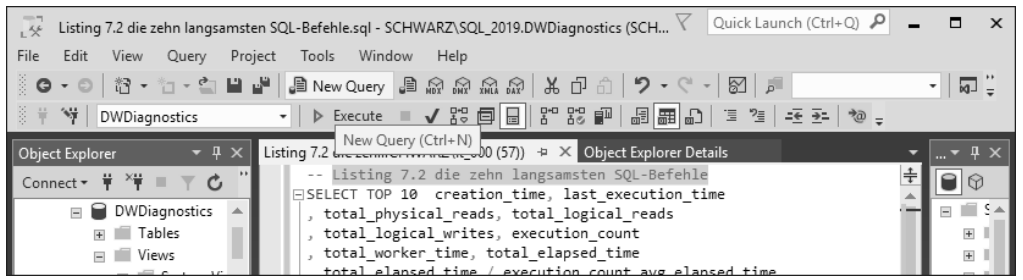

Abbildung 7.17 Schaltfläche »Neue Abfrage« (»New Query«) in der SSMS-Toolbar

Das SSMS öffnet daraufhin eine neues *.sql*-Dateifenster – auch Abfrage-Editor-Fenster genannt –, in das Sie T-SQL-Befehle eingeben werden. Den in das Abfrage-Editor-Fenster eingegebenen T-SQL-Code können Sie, wie in Microsoft-Programmen üblich, abspeichern und bei Bedarf wieder öffnen. Im Management Studio gespeicherte Skripte erhalten die Dateierweiterung *.sql*. Eine *.sql*-Datei ist eine normale Textdatei und kann auch mit Notepad oder einem anderen Texteditor bearbeitet werden. Durch ei-

nen Klick auf die Schaltfläche Ausführen (Execute) in der Toolbar führen Sie das aktive, im Abfrage-Editor-Fenster geladene SQL-Skript aus.

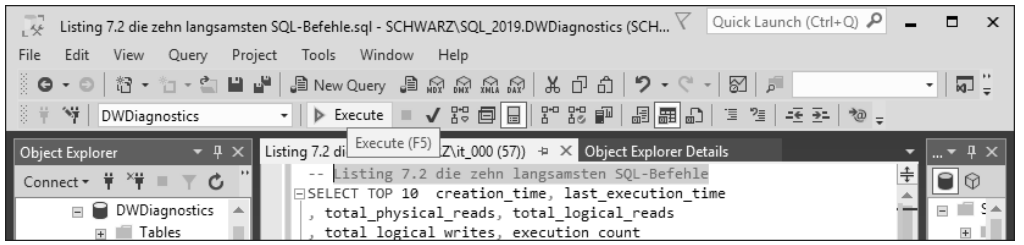

Abbildung 7.18 Die Schaltflächen »Ausführen« (»Execute«)

Die Funktionalität der Schaltflächen rechts neben der Ausführen-Schaltfläche (Execute) wird in Tabelle 7.1 erläutert.

Schaltfläche	Funktion	Beschreibung
■	Ausführung der Abfrage abbrechen (Cancel Executing Query)	Senden einer Abbruchanforderung an den Server, z. B. das Beenden von Abfragen, die übermäßig viel Zeit in Anspruch nehmen Tastaturkürzel: Alt+Break
Debug	Debuggen der Abfrage	Der Transact-SQL-Debugger wird aktiviert (in der aktuellen Version des Management Studios noch nicht vorhanden)
✓	Analysieren (Parse)	Analysieren des ausgewählten oder des gesamten T-SQL-Codes auf syntaktische Korrektheit hin – keine Ausführung Tastaturkürzel: Ctrl+F5
	geschätzten Ausführungsplan anzeigen (Display Estimated Execution Plan)	grafische Anzeige des geschätzten Ausführungsplans Tastaturkürzel: Ctrl+L

Tabelle 7.1 Funktionen der Schaltflächen in der SSMS-Toolbar

Kleiner Trick zum Ausführen von einzelnen T-SQL-Skriptbefehlen

Wenn Sie innerhalb eines T-SQL-Skripts einen Befehl mit der Maus markieren und anschließend auf Ausführen (Execute) klicken, wird nur dieser Befehl und *nicht* das ganze Skript ausgeführt.

Listing 7.3 zeigt Ihnen ein einfaches T-SQL-Skriptbeispiel, das die Zeilenanzahl der Tabelle »DemoTabelle« überprüft, abhängig von der Zeilenanzahl einen entsprechenden Text zusammensetzt und diesen anschließend in eine Logtabelle namens »LogTabelle« schreibt.

```
/* Einfaches T-SQL Skriptbeispiel */
-- Variablendeklaration
DECLARE @AnzahlZeilen INT;
DECLARE @LogText VARCHAR(200) = '';
-- Ermitteln der Zeilenanzahl der Tabelle DemoTabelle
-- und Schreiben in die Variable @AnzahlZeilen
SELECT @AnzahlZeilen = COUNT(*) FROM DemoTabelle;
-- Entscheidung mittels IF
IF (@AnzahlZeilen > 10)
BEGIN
-- Zuweisen einer Zeichenkette zur Variable @Logtext
SELECT @LogText =
 'Achtung, die Anzahl der Zeilen in der Tabelle DemoTabelle ist groesser ' +
CAST(@AnzahlZeilen AS VARCHAR(100));
END
ELSE
BEGIN
SELECT  @Logtext =
 'Achtung die Anzahl der Zeilen in der Tabelle DemoTabelle ist kleiner ' +
CAST(@ANZAHLZEILEN AS VARCHAR(100));
END
INSERT INTO LogTabelle (LogText) VALUES(@LogText);
```

Listing 7.3 Einfaches T-SQL-Skript

In Listing 7.3 erkennen Sie, dass in diesem T-SQL-Skript Variablen deklariert und zugewiesen, Entscheidungen getroffen und Datentypen konvertiert werden. Sie sehen in diesem Skript auch sogenannte *Kommentare (Comments)*. Kommentare ermöglichen es, zu den Skriptbefehlen erklärenden Text zu schreiben, der nicht als Befehl angesehen wird. Er dient dem Skriptentwickler als Notiz, um sich im Skriptcode besser zurechtzufinden.

Art des Kommentars	Beispiel
einzeiliger Kommentar	`-- Einzeiliger Kommentar`
mehrzeiliger Kommentar	`/* Das ist ein Kommentar,` `der über mehrere Zeilen geht */`

Tabelle 7.2 Ein- und mehrzeilige Kommentare

Hinweis

Zu meinem Glück habe ich es mir schon vor langer Zeit angewöhnt, die Ausgabe jedes Skript, das ich ausgeführt habe, als Kommentar hinten anzufügen und erneut zu speichern. So kann ich immer sehen, dass dieses Skript bereits ausgeführt wurde. Außerdem zeigt mir der Zeitstempel der Datei, wann das Skript ausgeführt wurde.

Alles in allem ist T-SQL eine sehr mächtige Sprache, mit der sich komplette Datenbankapplikationen entwickeln lassen. Es würde den Rahmen dieses Buches bei weitem überschreiten, hier T-SQL in seiner Komplexität zu behandeln. Möchten Sie tiefer in die Materie einsteigen, empfehlen wir Ihnen die Microsoft-Dokumentation.

7.3 Datenbanken anlegen

Eine Datenbank besteht immer aus mindestens zwei Dateien: der Datendatei und der Protokolldatei.

1. In der Datendatei (*Data File*) befinden sich die gespeicherten Daten der Datenbank. Die Dateierweiterung ist *.mdf*.
2. Die Transaktionsprotokolldatei (Transaction Log File) sammelt die Informationen zu den durchgeführten Transaktionen, das heißt den Änderungen am Datenbestand. Die Dateierweiterung ist *.ldf*.

Eine Datenbank kann aber auch aus mehr als nur den zwei genannten Dateien bestehen. Es kann mehrere Datendateien und Transaktionsprotokolldateien geben, wobei die erste Datei immer die sogenannte *primäre (primary)* Datendatei ist und jede weitere Datendatei als *sekundäre (secondary)* Datendatei bezeichnet wird.

In der primären Datendatei sind neben den eigentlichen Daten Startinformationen für die Datenbank sowie Verweise auf die anderen Dateien der Datenbank gespeichert.

Wie bereits erwähnt, hat die *primäre* Datendatei die Dateierweiterung *.mdf*, jede weitere Datendatei sollte die empfohlene Erweiterung *.ndf* haben. Die Dateierweiterung des Transaktionsprotokolls sollte immer *.ldf* sein. Wie Sie weitere Datendateien einer Datenbank hinzufügen, lesen Sie in Abschnitt 7.3.2, »Dateigruppen – Strategie zur Verteilung von Objekten auf Datenträgern«.

Hinweis zu den Dateierweiterungen

SQL Server zwingt Sie nicht, die empfohlenen Dateierweiterungen zu verwenden, allerdings sollten Sie sich an die empfohlenen Vorgaben halten, um klar erkennen zu können, um welche Art von Datei es sich handelt.

Eine Datenbank in SQL Server anzulegen, kann einfach oder komplex sein; das hängt im Wesentlichen davon ab, inwieweit Sie für die Erstellung der Datenbank von den jeweiligen Standardeinstellungen – z. B. Speicherort der Datenbankdateien – abweichen.

Im folgenden Beispiel erläutern wir Ihnen die einfache Variante des Anlegens einer Datenbank namens »demo«:

1. Öffnen Sie das SSMS. Verbinden Sie sich mit der Instanz von SQL Server, auf der Sie eine Datenbank anlegen möchten.
2. Klicken Sie mit der rechten Maustaste im Objekt-Explorer auf den Ordner Datenbanken (Databases), und wählen Sie Neue Datenbank (New Database).
3. Im daraufhin erscheinenden Dialog geben Sie im Feld Datenbankname (Database Name) den gewünschten Namen – hier »demo« – der Datenbank an.
4. Klicken Sie auf OK, und die Datenbank wird erstellt und anschließend im Objekt-Explorer angezeigt.

Sie können eine Datenbank mit dem Namen »demo« auch anlegen, indem Sie direkt den T-SQL-Befehl `CREATE DATABASE demo`; im Abfrage-Editor-Fenster des SSMS eingeben und ausführen.

7.3.1 Was geschieht beim Erstellen einer Datenbank?

Für den Befehl zum Erzeugen einer Datenbank, `CREATE DATABASE`, können Sie eine Vielzahl von Parametern setzen. Geben Sie diese Parameter nicht an, so wie im Beispiel zuvor, verwendet SQL Server entsprechende Standardeinstellungen (*Default Settings*) zum Erzeugen der Datenbank.

Ein wichtiger Punkt ist die Information, wo denn überhaupt die Dateien, in denen die Daten gespeichert werden, auf der Festplatte abgelegt werden sollen. SQL Server legt Ihre Datenbank in den für SQL Server schon bei der Installation festgelegten Standardverzeichnissen an. Dies gilt natürlich nur, wenn Sie den Pfad nicht gesondert bei der Datenbankerstellung angeben.

Die in SQL Server hinterlegten Standardspeicherorte dieser Dateien können Sie sich wie folgt anzeigen lassen:

1. Öffnen Sie das SSMS. Verbinden Sie sich mit der Instanz von SQL Server, auf der Sie eine Datenbank anlegen möchten.
2. Klicken Sie mit der rechten Maustaste auf den Serverknoten, und wählen Sie im zugehörigen Kontextmenü den Menüpunkt Eigenschaften (Properties) und dann der Reiter Datenbankeinstellungen (Database Settings) aus.
3. Im sich öffnenden Dialog finden Sie im Bereich Standardspeicherorte für Datenbank (Database default locations) die Dateipfade, in denen Ihre Datendateien und Protokolldateien angelegt werden.

Jede Datenbank kann Daten und Protokolldateien an anderen Orten gespeichert haben. Um sich die Dateispeicherpfade einer Datenbank anzeigen zu lassen, klicken Sie mit der rechten Maustaste auf die Datenbank und wählen Eigenschaften (Properties) aus. Im Dialog klicken Sie auf Dateien (Files). In der Tabelle Datenbankdateien (Database Files) finden Sie alle Dateien, aus denen die Datenbank besteht, inklusive der Verzeichnisse, die Sie in der Spalte Pfad (Path) sehen.

Hinweis zur Vergabe von Datenbanknamen

- Der Datenbankname kann bis zu 128 Zeichen lang sein.
- Er muss eindeutig sein.
- Sie sollten es vermeiden, Leerzeichen oder Sonderzeichen im Namen zu verwenden. Wenn Sie dies trotzdem tun möchten oder müssen, setzen Sie den Datenbanknamen in eckige Klammern, damit er akzeptiert wird. Außerdem sollten Sie sich, wenn möglich, bei den Sonderzeichen auf den Unterstrich beschränken. Er funktioniert mit den meisten Anwendungen ohne Schwierigkeiten.

7.3.2 Dateigruppen – Strategie zur Verteilung von Objekten auf Datenträgern

Wie wir bereits erläutert haben, besteht eine SQL-Server-Datenbank aus mindestens einer Datendatei und einer Transaktionsprotokolldatei. Es gibt noch eine weitere Kategorisierung innerhalb einer Datenbank: die Dateigruppe. Eine *Dateigruppe* (*File Group*) ist eine Verwaltungseinheit, in der eine oder mehrere Datendateien logisch kategorisiert und unter dem Namen der Dateigruppe zusammengefasst sind. Dateien, die logisch einer Dateigruppe zugeordnet sind, können, obwohl sie zu derselben Dateigruppe gehören, auf unterschiedlichen Festplatten oder Speicherorten liegen.

Innerhalb einer Datenbank gibt es immer eine Standarddateigruppe mit dem Namen »PRIMARY«. Diese Dateigruppe enthält die primäre Datendatei und auch andere sekundäre Datendateien, die dieser Gruppe zugeordnet wurden.

Wenn Sie in einer Datenbank ein Datenbankobjekt (wie z. B. eine Tabelle, einen Index oder eine Sicht) ohne Angabe der zugehörigen Dateigruppe erstellen, wird dieses Datenbankobjekt immer in den Datendateien der Standarddateigruppe gespeichert. Sie können bei Bedarf auch eine eigene benutzerdefinierte Dateigruppe anlegen und zur Standarddateigruppe erklären. Genaueres zum Speichern von Datenbankobjekten in Dateigruppen lesen Sie im folgenden Abschnitt.

Wozu sind Dateigruppen gut?

Erstellen Sie innerhalb einer Datenbank Datenbankobjekte wie Tabellen (tables), gespeicherte Prozeduren (stored procedures) oder Sichten (views), können Sie angeben, in welcher Dateigruppe dieses Datenbankobjekt gespeichert werden soll.

Auf diese Art und Weise können Sie steuern, dass z. B. Tabellen und Indizes, auf die sehr häufig lesend zugegriffen wird, auf einem Datenträger liegen, der schnelle Lesezugriffe ermöglicht. Tabellen, auf die sehr selten zugegriffen wird, legen Sie auf weniger schnelle Datenträger.

Der Befehl `CREATE TABLE adressen (ID INT, Nachname VARCHAR(100),Vorname VARCHAR(100)) ON dateigruppe1` speichert die Tabelle `adressen` in den Datendateien, die der `dateigruppe1` angehören. Dies setzt natürlich voraus, dass die Dateigruppe mit Namen `dateigruppe1` überhaupt existiert und dieser Dateigruppe auch Datendateien zugeordnet sind.

Anlegen einer Dateigruppe mit SQL Server Management Studio

1. Klicken Sie mit der rechten Maustaste auf die Datenbank, in der Sie die Dateigruppe anlegen möchten, und wählen Sie den Menüpunkt Eigenschaften (Properties) aus dem Kontextmenü aus.
2. Wählen Sie aus dem Bereich Seite auswählen (Select a page) den Punkt Dateigruppen (Filegroups).
3. Klicken Sie auf die Schaltfläche Hinzufügen (Add Filegroup), und geben Sie in der neu angelegten Zeile der Tabelle in der Spalte Name »dateigruppe1« ein (siehe Abbildung 7.19).
4. Bestätigen Sie dies mit OK, um die Dateigruppe anzulegen.

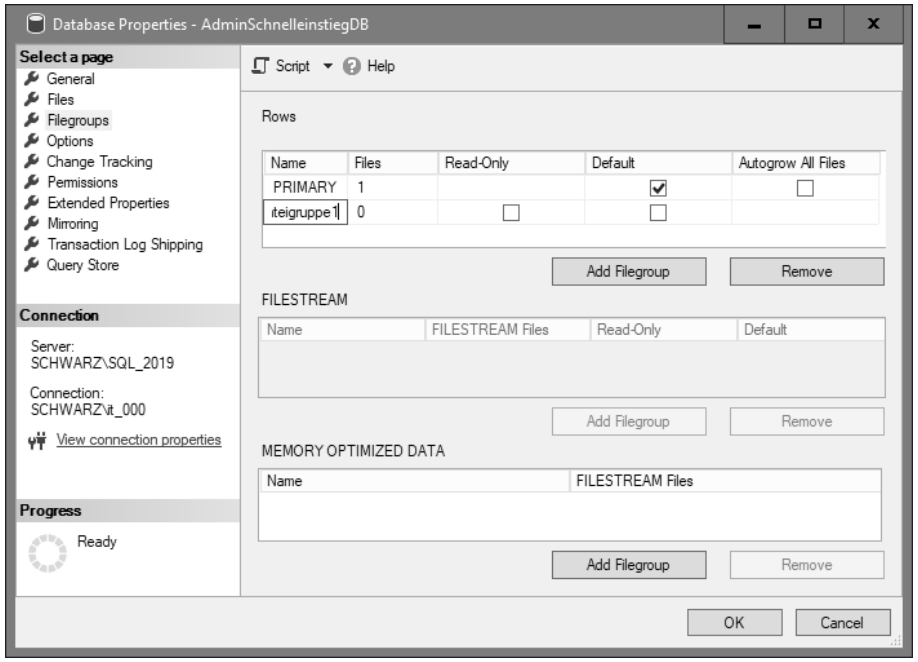

Abbildung 7.19 Anlegen einer Dateigruppe

Die Dateigruppe wurde jetzt in der Datenbank zwar angelegt, hat jedoch noch keine Funktion im Datengefüge der Datenbank. Um Datenbankobjekte wie Tabellen oder Sichten der Dateigruppe hinzufügen zu können, müssen Sie der Dateigruppe eine Datendatei der Datenbank zuweisen.

Hinzufügen einer Datendatei, die der Dateigruppe »dateigruppe1« angehört

1. Klicken Sie mit der rechten Maustaste auf die Datenbank, in der Sie die Datendatei anlegen möchten, und wählen Sie den Menüpunkt Eigenschaften (Properties) aus dem Kontextmenü aus.
2. Wählen Sie aus dem Bereich Seite auswählen (Select a page) den Punkt Dateien (Files).
3. Klicken Sie auf die Schaltfläche Hinzufügen (Add), und geben Sie in der neu angelegten Zeile der dargestellten Tabelle in der Spalte Logischer Name (Logical Name) »datendatei_neu« »New File« ein.

Was ist der logische Dateiname?

Der *logische* Dateiname ist eine Bezeichnung, die dazu dient, innerhalb von SQL-Anweisungen auf die Datei verweisen zu können, ohne jeweils den kompletten Pfad angeben zu müssen, also eine Art Aliasname, der die Datendatei repräsentiert.

4. In der Spalte Dateityp (File Type) wählen Sie Zeilendaten (ROWS Data) aus, was bedeutet, dass Sie eine Datendatei zum Speichern der Daten und keine Protokolldatei anlegen möchten.
5. In der Spalte Dateigruppe (Filegroup) bestimmen Sie nun, welcher Dateigruppe die Datendatei angehören soll. Hier finden Sie die von Ihnen angelegten Dateigruppen sowie die Dateigruppe PRIMARY. Wählen Sie die zuvor angelegte Dateigruppe DATEIGRUPPE1 aus.
6. Legen Sie jetzt in der Spalte Pfad (Path) den Speicherort fest, an dem Ihre Datendatei liegen soll, und klicken Sie auf OK.

Sie könnten jetzt weitere Datendateien anlegen und diese wiederum der Dateigruppe oder anderen Dateigruppen zuweisen.

Merke

- Sie können Datenbankobjekte nur Dateigruppen, nicht den Datendateien der Datenbank direkt zuweisen.
- Es gibt immer eine primäre Dateigruppe mit dem Namen PRIMARY.
- Sie können pro Datenbank bis zu 256 Dateigruppen anlegen.
- Protokolldateien können Sie keiner Dateigruppe zuordnen.

Der Einsatz von Dateigruppen und die Zuweisung von Datenbankobjekten zu Dateigruppen bieten folgende Vorteile:

- Sie können steuern, auf welchen physischen Datenträgermedien Datenobjekte gespeichert werden. Bei einer Verteilung auf mehrere physische Speicherorte ist SQL Server aufgrund von paralleler Verarbeitung in der Regel schneller.
- Sie können die Speicherkapazität einer Datenbank durch Hinzufügen einer Datendatei, die auf einer neuen Festplatte gespeichert wird, flexibel erhöhen.

7.3.3 Fehler finden – Debuggen von T-SQL

Bei der Entwicklung von T-SQL-Skripten kommt es vor, dass das Skript Fehler enthält, Programmabläufe unübersichtlich sind oder Sie in Ausnahmesituationen Fehler in der Skriptfunktionalität erst während der Testphase entdecken. Da ein Skript aus den unterschiedlichsten Operatoren, Variablen, Programmflusskomponenten und SQL-Befehlen besteht, ist es sehr schwierig, in einem solchen Fall ohne Hilfsmittel diese Fehler ausfindig zu machen.

Um in einem T-SQL-Skript Fehler zu entdecken und zu analysieren, bietet Management Studio die Möglichkeit, T-SQL-Skripte zu debuggen. So können Sie die einzelnen Skriptbefehle einzeln ausführen und die Inhalte von Variablen und Programmabläufe dieses Skripts analysieren. Zudem ermöglicht das Setzen von Haltepunkten, an einem bestimmten Punkt das Skript anzuhalten und das Analysieren der Befehle und Variablen an diesem Punkt zu beginnen.

> **Hinweis**
>
> Im aktuellen ManagementStudio (ab Version 18) gibt es leider keinen eigenständigen T-SQL-Debugger mehr. Sie müssen sich auf ihre Fähigkeiten und die rote »Schlangenlinie« unter fehlerhaftem Code verlassen. Sie können mit vielen `SELECT` oder `PRINT` Anweisungen ihre Scripts auch selbst debuggen.

Den Debugger für ein im SSMS geöffnetes Skript starten Sie, indem Sie in der Toolbar auf die Schaltfläche Debuggen (Debug) klicken (siehe Abbildung 7.20).

Abbildung 7.20 Schaltfläche »Debuggen« im SSMS

Befindet sich das Management Studio im Debugging-Modus, wird in der Toolbar die Symbolleiste Debuggen (Debug) eingeblendet, die Sie in Abbildung 7.21 sehen.

Abbildung 7.21 Symbolleiste »Debuggen«

Die Symbolleiste Debuggen (Debug) enthält unter anderem die Schaltflächen Einzelschritt (Step Into), Prozedurschritt (Step Over) und Debugging beenden (Stop Debugging). Durch einen Klick auf Einzelschritt (Step Into) springt der Befehlszeiger im linken Bereich des T-SQL-Skriptfensters auf den nächsten auszuführenden Befehl. Der Befehlszeiger wird in Form eines gelben Pfeils am linken Rand des Abfrage-Editor-Fensters dargestellt (siehe Abbildung 7.22).

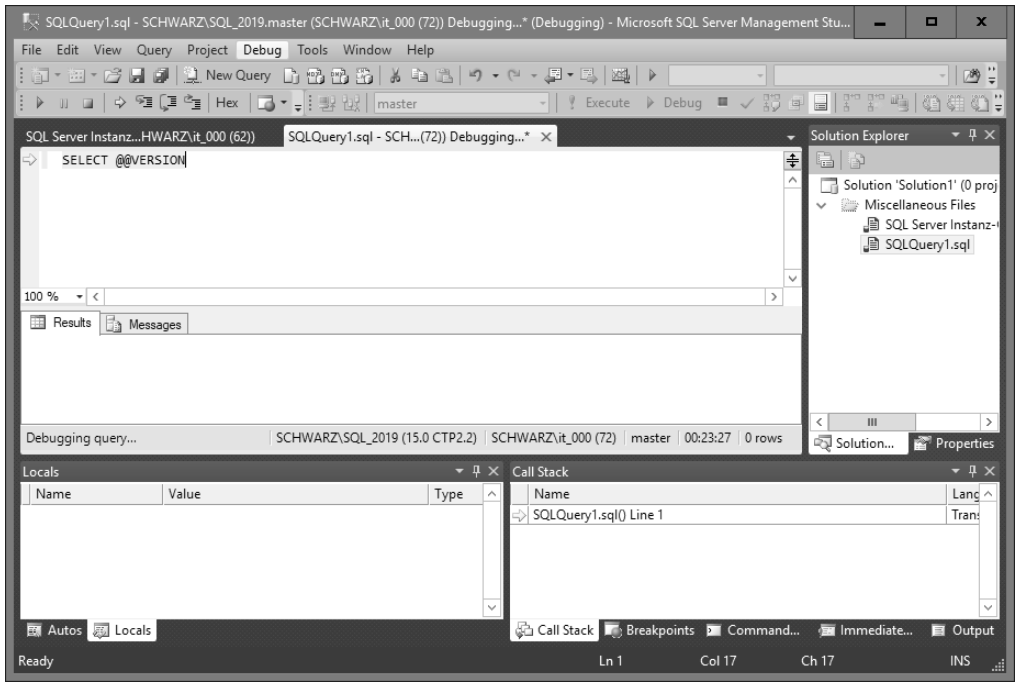

Abbildung 7.22 Befehlszeiger im Debugging-Modus

Da ein Skript auch gespeicherte Prozeduren aufrufen kann, die als selbständige Skriptmodule fungieren und sehr umfangreich sein können, ist es nicht immer wünschenswert, auch diese Prozeduren zu debuggen. Dies geschähe allerdings automatisch, wenn der Befehlszeiger auf eine Anweisung trifft, die eine gespeicherte Prozedur aufruft. Mit der Schaltfläche Prozedurschritt (Step Over) führen Sie lediglich den Befehl aus und debuggen nicht in die gespeicherte Prozedur hinein.

Möchten Sie ein Skript debuggen, ist es natürlich auch wichtig, die Inhalte von Variablen zu analysieren. Zu diesem Zweck wird wie in Abbildung 7.23 im Debugging-Modus das Fenster Lokal (Locals) eingeblendet, das die im Skript enthaltenen Variablen und ihre Inhalte anzeigt.

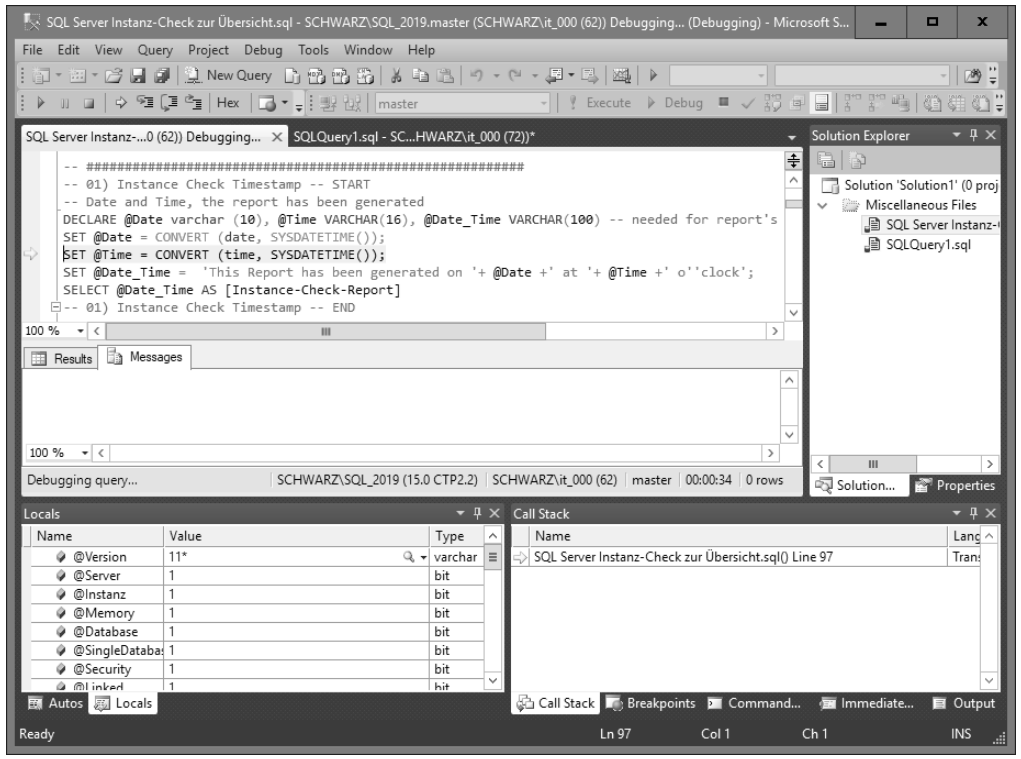

Abbildung 7.23 Fenster »Lokal« zur Analyse von Variablen im Debugging-Modus des SQL Server Management Studios

> **Anzeige der Debugging-Fenster**
>
> Sollte das hier beschriebene Fenster Lokal (Locals) nicht angezeigt werden, können Sie es im SSMS-Menü Debuggen (Debug) • Fenster (Windows) einblenden.

Die Debugging-Funktionalität im Management Studio ist sehr hilfreich und ermöglicht auch das Debuggen von Triggern, das Auffinden von Fehlern und die Analyse von Skriptcode innerhalb eines annehmbaren Zeitraums.

7.4 T-SQL – Die wichtigsten Befehle für den Administrator

7.4.1 DML – SELECT, INSERT, UPDATE, DELETE

Datenbanken haben eine wichtige Aufgabe. Daten müssen effizient gespeichert werden können. In der Regel werden Daten von Datenbankanwendungen gelöscht, geändert, oder es werden neue Daten hinzugefügt. Die Datenqualität von Daten definiert sich dadurch, dass in einer Datenbank gespeicherte Daten den Anforderungen entsprechend ausgewertet und natürlich auch bearbeitet werden können. Kurz gesagt: Daten in einer Datenbank sind nur gut, wenn Sie sie auch wieder herausholen, performant auswerten und bearbeiten können.

Zu diesem Zweck gibt es in SQL vier wichtige Befehle, die in Tabelle 7.3 aufgelistet sind.

Befehl	Beschreibung
SELECT	Suchen, Filtern und Anzeigen von Daten
INSERT	Hinzufügen von Daten
UPDATE	Aktualisieren von Daten
DELETE	Löschen von Daten

Tabelle 7.3 Die vier wichtigsten SQL-DML-Befehle

Diese vier Befehle sind die grundlegenden Befehle zum Manipulieren von Daten innerhalb einer Datenbank. Aus diesem Grund werden diese Befehle auch als *Data Manipulation Language* (*DML*) bezeichnet. Etwas strittig ist hierbei natürlich der SELECT-Befehl, der eigentlich keine Daten manipuliert, sondern lediglich anzeigt. Dennoch wird häufig auch dieser Befehl im Zusammenhang mit DML aufgeführt. Vielfach wird der SELECT-Befehl auch einer weiteren Kategorie zugeordnet, der *Data Query Language* (*DQL*).

Tabelle 7.4 zeigt einige Beispiele für die genannten Befehle.

Geplante Aktion	Code
Anzeigen aller Daten und Spalten der Tabelle »DemoTabelle«	`SELECT * FROM DemoTabelle;`
Anzeigen aller Zeilen und Spalten der Tabelle »DemoTabelle«, die in der Spalte *Nachname* den Wert »Franke« gespeichert haben	`SELECT * FROM DemoTabelle WHERE Nachname = 'FRANKE'`
Löschen aller Daten aus der Tabelle »DemoTabelle«	`DELETE FROM DemoTabelle`
Löschen nur der Datensätze, die innerhalb der Spalte *Nachname* den Wert »Franke« gespeichert haben	`DELETE FROM DemoTabelle WHERE Nachname = 'FRANKE'`
Ändern des Nachnamens aller Zeilen in »Friedrich«, die in *Nachname* »Franke« stehen haben	`UPDATE DemoTabelle SET Nachname 'FRIEDRICH' WHERE Nachname = 'FRANKE'`

Tabelle 7.4 Geplante Aktion und entsprechender SQL-Code

7.4.2 DDL – CREATE, ALTER, DROP

Wenn Sie Befehle an SQL Server senden, die die Struktur der Datenbank oder der Datenbankobjekte ändern, spricht man von Befehlen der sogenannten *Data Definition Language* (*DDL*). Im Wesentlichen gibt es drei Befehle zum Erzeugen, Ändern und Löschen von Datenstrukturen. Sie finden sie in Tabelle 7.5.

Befehl	Beschreibung
CREATE	Anlegen von Strukturen
ALTER	Ändern von Strukturen
DROP	Löschen von Strukturen

Tabelle 7.5 Erzeugen, Ändern und Löschen – die Befehle

Um Strukturen anzulegen, verwenden Sie in der Regel den Befehl CREATE. CREATE DATABASE, CREATE TABLE, CREATE PROCEDURE und CREATE TRIGGER legen die unterschiedlichen Datenbankobjekte an. Die gleichen Strukturen können Sie mit der Anweisung ALTER ändern und mit DROP löschen.

In Tabelle 7.6 sehen Sie einige Beispiele zu den einzelnen Befehlen.

Konsolenbefehl	Beschreibung
CREATE DATABASE Demo;	Legt die Datenbank mit Namen Demo an.
DROP DATABASE Demo;	Löscht die Datenbank Demo.
CREATE TABLE DemoTabelle (ID INT IDENTITY, Nachname NVARCHAR(50), Vorname NVARCHAR(50));	Legt eine Tabelle mit drei Spalten an. Spalte 1 enthält eine ganze Zahl vom Datentyp Integer, die mit jedem Datensatz automatisch um 1 erhöht wird. Die Spalten 2 und 3 ermöglichen die Eingabe von Text von jeweils 50 Zeichen Länge.
ALTER TABLE DemoTabelle ADD Strasse NVARCHAR(100);	Hinzufügen der Spalte Strasse mit einer möglichen Länge von 100 Zeichen
DROP TABLE [dbo].[DemoTabelle]	Löschen der Tabelle DemoTabelle im Schema »dbo«

Tabelle 7.6 Erzeugen, Ändern und Löschen – Beispiele

Kapitel 13
Skalierbarkeit von SQL Server

Skalierung ist ein wichtiges Thema, um Investitionen für die Zukunft zu sichern. Wenn Unternehmen wachsen, ist dasjenige Unternehmen im Vorteil, das keine völlig neuen Lösungen implementieren muss, sondern nur weitere Lizenzen kauft.

Was bedeutet *Skalierung*? In welchen Szenarien ist es wichtig, über Skalierung nachzudenken, und können Sie das Thema schon bei der Planung berücksichtigen?

Es gibt zwei Arten der Skalierung: Scale-up und Scale-out

Das *Scale-up* bedeutet den geringsten Aufwand auf Anwendungsseite. Hier werden »nur« die Leistungsträger der unterliegenden Hardware verstärkt. Mehr RAM, weitere CPUs oder auch eine Umstellung von klassischen Festplatten auf SSDs sind die Faktoren, allerdings auch die Grenze. Ist das Maximum verbaut, dann ist kein weiteres Scale-up mehr möglich.

Beim *Scale-out* werden Dienste, Instanzen oder/und Datenbanken auf zusätzliche Hardware ausgelagert. Dabei ist der Aufwand für den Umbau größer, dafür sind die Möglichkeiten der Skalierung nahezu unbegrenzt.

SQL Server zu skalieren, ist in vielerlei Hinsicht ein wichtiges Thema. Es betrifft z. B. Administratoren, die aus Leistungsgründen SQL Server skalieren müssen oder die eine hohe Verfügbarkeit von SQL Server sicherstellen möchten, oder Entwickler, die ein Framework wie Service Broker nutzen. Der Vorteil einer Skalierung liegt auf der Hand: Sie können zunächst mit der Installation einer einzigen Instanz von SQL Server beginnen, der dann weitere Instanzen folgen. Muss skaliert werden, können Sie Dienste so auf weitere Server verlagern oder, im Fall von Service Broker, die Routen zu den neuen Servern ändern.

SQL Server wird kostenlos als Express Edition angeboten. Unternehmen beginnen oft mit dieser Version. Sie stellen jedoch nach einiger Zeit fest, dass es weitere interessante Dienste wie die Integration Services für ETL-Prozesse gibt, und haben den Wunsch, diese zu verwenden, um ihre heterogene IT-Landschaft zu verbinden und vielleicht zu konsolidieren. Letztendlich erfolgt entweder ein Upgrade der vorhandenen Version, oder es kommen weitere Installationen von SQL Server hinzu. Das Upgrade ist einfach; Sie können kostenlos beginnen und später auf eine lizenzpflich-

tige Version aktualisieren. Eine Alternative ist es, nicht zu aktualisieren, sondern weitere Installationen hinzuzufügen und Dienste von SQL Server auf unterschiedlichen Rechnern zu installieren.

13.1 Verteilen der SQL-Server-Dienste

SQL Server besteht aus den bereits in Kapitel 5, »Konfigurieren von SQL Server«, genannten Diensten, die sich in einer Infrastruktur gut verteilen lassen. Ein komplexes Datenbanksystem, das viele Transaktionen verarbeitet, muss oft aus Leistungsgründen auf einem eigenen Server ausgeführt werden. Die *Integration Services*, die bis dahin vielleicht lediglich Wartungspläne ausführen, laufen auf demselben Server wie das Datenbankmodul. Ihr Unternehmen plant vielleicht in Zukunft die Einführung eines Data Warehouses. Komplexe Analysen sollen aus dem Data Warehouse erfolgen, und ein Berichtswesen soll eingeführt werden.

Es zeigt sich dann häufig bereits sehr früh, dass die *Analysis Services* und die *Reporting Services* aus Leistungsgründen auf weiteren separaten Servern ausgeführt werden müssen. Sie sehen also: Gerade hier sind Sie sehr schnell beim Thema Skalierung angekommen, denn in der Praxis erfüllt selten ein einzelner Server diese komplexen Anforderungen. Allein das Thema in Kapitel 22, »Parallel Data Warehouse (PDW)«, zeigt, wie schnell heute bei sehr großen Datenbanken skaliert werden muss.

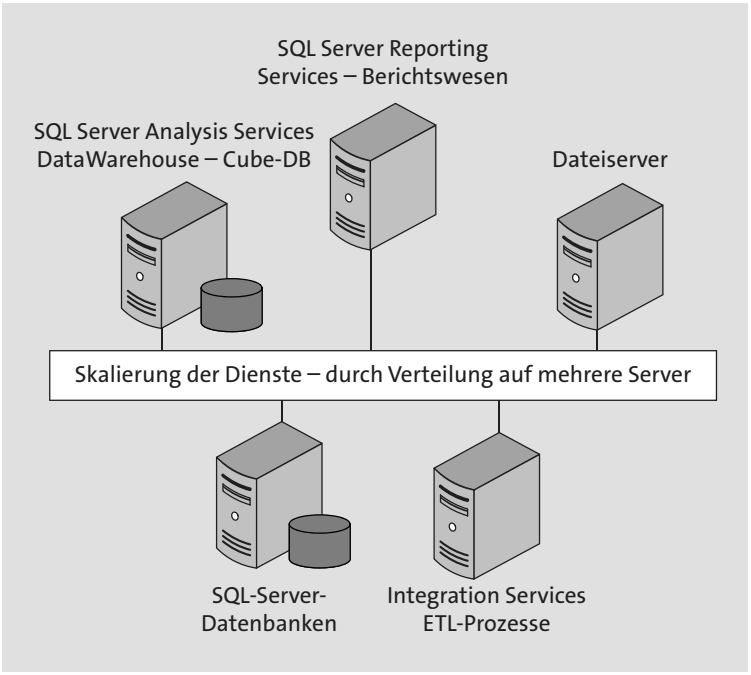

Abbildung 13.1 Verteilung der SQL-Server-Dienste auf mehrere Server

13.2 SQL Server und NLB-Cluster

Ein Netzwerklastenausgleich-Cluster (*NLB-Cluster* oder *Network-Load-Balancing-Cluster*) kann seit Windows Server 2008 aufgebaut werden und hat die Aufgabe, Anfragen von Clients z. B. an eine Datenbank zu verteilen. Somit entspricht der NLB-Cluster einer typischen Lastenverteilung. Das kann in den verschiedensten Szenarien hilfreich sein, wenn es darum geht, die Last auf einem einzelnen Server zu reduzieren und auf mehrere Server zu verteilen. Sowohl beim Datenbankmodul als auch bei den Reporting Services – oder, wie Sie im nächsten Abschnitt noch sehen werden, bei den Analysis Services – ist das eine Option zur Skalierung.

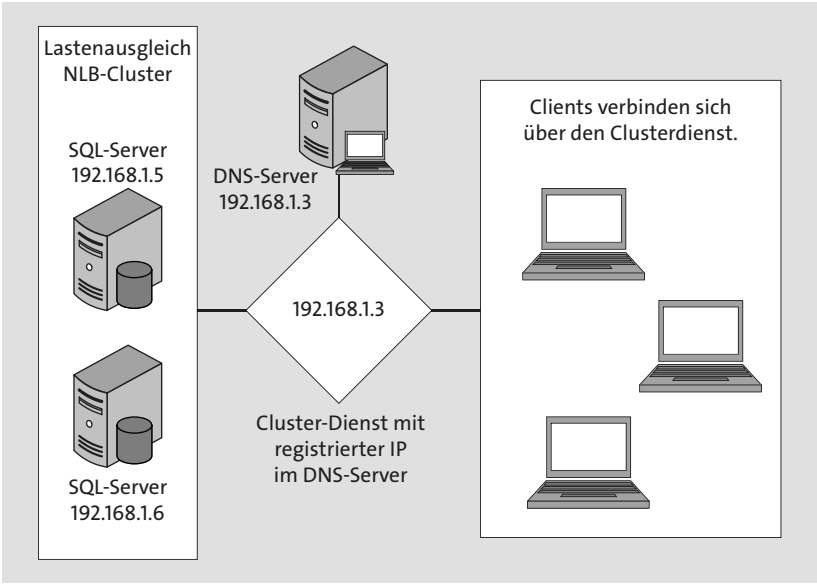

Abbildung 13.2 Netzwerklastenausgleich zur Verteilung der Clientanfragen

13.2.1 Reporting Services und Lastenausgleich

Abbildung 13.3 zeigt, wie der Berichtsserver innerhalb eines NLB-Clusters ausgeführt wird. Die Berichtsserver-Datenbank liegt außerhalb des Clusters und muss hier nicht redundant vorhanden sein. Hier werden lediglich die Anfragen der Clients über das Lastenausgleichsmodul an den Berichtsserver verteilt. Dabei wird eine virtuelle IP-Adresse im Domain Name System (DNS) registriert und steht den Clientanwendungen bei Anfragen zur Verfügung. Somit entscheidet das Lastenausgleichsmodul, zu welchem Server die Anfragen der Anwendung weitergeleitet werden.

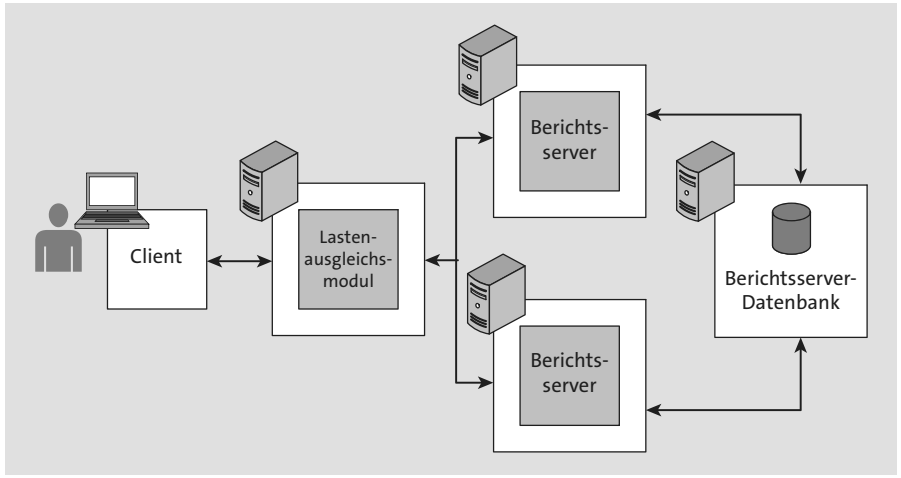

Abbildung 13.3 Ein NLB-Cluster für Reporting Services

13.2.2 Failover-Cluster

Eine weitere Möglichkeit besteht darin, zusätzlich zum NLB-Cluster für den Berichtsserver die Datenbank der Reporting Services in einem Failover-Cluster unterzubringen. Die technologischen Hintergründe zum Failover-Cluster und dem Protokollversand werden wir Ihnen in Kapitel 15, »Hochverfügbarkeitslösungen«, vermitteln. An dieser Stelle ist es lediglich wichtig zu verstehen, dass mit dem Failover-Cluster der Ausfall des SQL Servers und somit der Berichtsserver-Datenbank verhindert wird. Bei einem Ausfall übernimmt ein zweiter, redundanter SQL Server. Kommt es jetzt zu einem Ausfall einer Instanz von SQL Server, wird durch *Failover* auf die zweite Instanz geschwenkt, und die Datenbank steht weiter zur Verfügung.

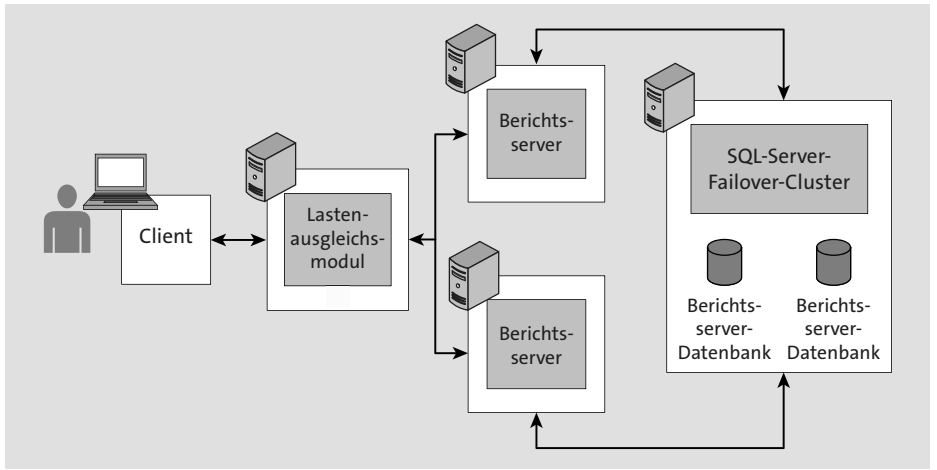

Abbildung 13.4 Kombination von NLB-Cluster und Failover-Cluster

In Abbildung 13.4 ist sehr gut zu erkennen, wie Sie eine Kombination aus Lastenausgleich und Failover-Cluster sinnvoll einsetzen können, um sowohl die Leistung als auch die Verfügbarkeit zu erhöhen.

13.2.3 SQL-Server-Protokollversand (Transaction Log Shipping)

In Abbildung 13.5 sehen Sie ein Beispiel dafür, wie ein NLB-Cluster in Verbindung mit dem Protokollversand eingesetzt wird. Dabei wird die Möglichkeit genutzt, das Transaktionsprotokoll von SQL Server nach einer Sicherung zu versenden. Das versendete Transaktionsprotokoll wird über einen Auftrag im SQL Server Agent der sekundären Instanz wiederhergestellt.

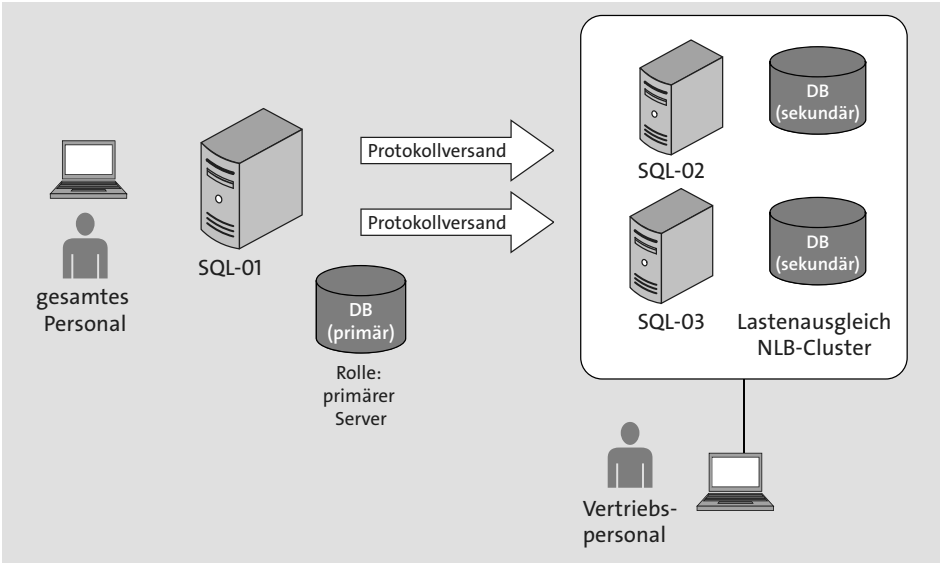

Abbildung 13.5 Kombination Protokollversand und NLB-Cluster

Das Beispiel beruht auf der Tatsache, dass beim Protokollversand mehrere Empfänger konfiguriert werden können und diese sich dann gemeinsam in einem NLB-Cluster befinden. Das ist beim Protokollversand unter Umständen sinnvoll, da auf die Daten der Protokollempfänger lesend zugegriffen werden kann. In diesem Fall erhalten die Empfänger vom primären Server jeweils in einem zeitlichen Intervall die Änderungen über das Transaktionsprotokoll. Eine Lastenverteilung sorgt hier für eine Reduzierung der Last auf den einzelnen Empfängern. Diese Empfänger können z. B. für Reporting-Zwecke genutzt werden. Anders als in einem Failover-Cluster sind beim NLB-Cluster die Datenbanken jeweils pro Knoten vorhanden. Auch hier zeigt sich, wie Sie mit Hilfe des Protokollversands SQL Server sehr gut skalieren können und es lediglich eines Hinzufügens weiterer Instanzen bedarf.

13.3 Skalierung der Analysis Services

Die Analysis Services werden wir in Kapitel 24 vorstellen. In diesem Abschnitt geht es uns in erster Linie darum, zu zeigen, welche Rolle die Analysis Services hinsichtlich einer Skalierung von SQL Server spielen. Um das Problem von langwierigen Abfragen auf ein Data Warehouse zu vermeiden, sollten Sie schon frühzeitig im Planungsprozess die Themen Design, Partitionierung und Skalierung der Analysis Services betrachten. Sie müssen sich jedoch im Klaren darüber sein, was Sie erreichen möchten. Die Analysis Services bieten einige Optionen für eine Optimierung oder Skalierung. Auf jeden Fall sollten Sie sich zuerst auf ein gutes Aggregations- und Partitionierungsdesign konzentrieren.

Wenn Ihr Performanceengpass die Prozessorauslastung auf einem einzigen System ist und eine sehr große Anzahl an Benutzern parallel Abfragen ausführt, können Sie die Abfrageleistung durch einen NLB-Cluster erhöhen. Wenn Ihre Benutzer viele Abfragen ausführen, die tatsächlich eine große Anzahl an Datenscans erfordern, ist die Wahrscheinlichkeit, dass eine Aggregation dies abdeckt, eher sehr gering. Ein NLB-Cluster kann hier ebenfalls eine gute Lösung sein. Seien Sie sich jedoch darüber im Klaren, dass die Datencaches auf jedem der Server in dem NLB-Cluster unterschiedlich sein werden, was zu Differenzen in der Abfrage-Antwort-Zeit von Abfrage zu Abfrage eines Benutzers führt.

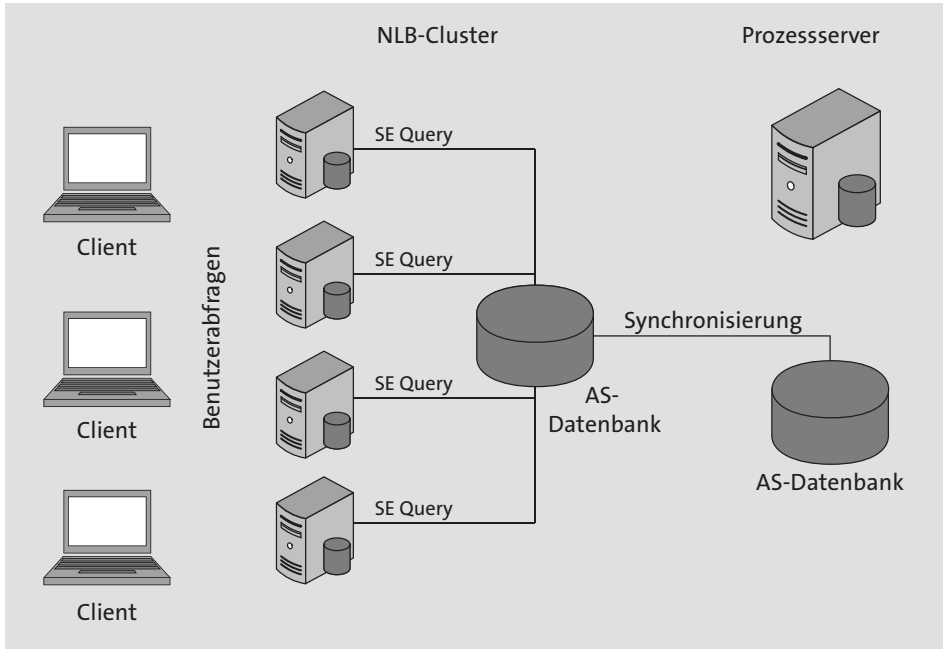

Abbildung 13.6 NLB-Cluster mit Nur-Lese-Servern

Ein NLB-Cluster kann auch verwendet werden, um die Verfügbarkeit zu erhöhen, falls ein Analysis-Server ausfällt. Eine zusätzliche Möglichkeit zur Erhöhung der Leistung mit Hilfe eines NLB-Clusters besteht in der Verteilung der Verarbeitungstasks auf einen Offlineserver. Wurden neue Daten über den Offlineserver verarbeitet, können Sie die Analysis-Server im Cluster mit Hilfe der Synchronisierung aktualisieren.

Abbildung 13.6 zeigt ein Szenario, bei dem Abfragen an einen NLB-Cluster gerichtet sind. Der Clusterservice entscheidet je nach Last, welcher Nur-Lese-Query-Server die Abfrage beantwortet.

Ein NLB-Cluster bedeutet nicht zwingend, dass Sie Ihre Performance erhöhen, wenn die Verarbeitung das Problem ist oder Sie eine einzelne Abfrage eines Users optimieren möchten. Beachten Sie auch, dass die Verwendung eines NLB-Clusters der Einschränkung unterliegt, dass ein Zurückschreiben nicht möglich ist.

13.4 Skalierbare freigegebene Datenbanken

Eine einfache Möglichkeit der Skalierung wird durch skalierbare freigegebene Datenbanken erreicht. Eine Datenbank wird als schreibgeschützt markiert, um sie dann von mehreren Analysis-Services-Instanzen zu nutzen. Das bedeutet: Mehrere Instanzen von Analysis Services können sich ein einzelnes Datenverzeichnis teilen, das sich z. B. auf einem *SAN* befindet. Grundsätzlich entspricht das der klassischen Shared-Disk-Lösung. Da die Analysis Services und die Reporting Services lesend auf Daten zugreifen, gibt es keine Probleme mit Sperren.

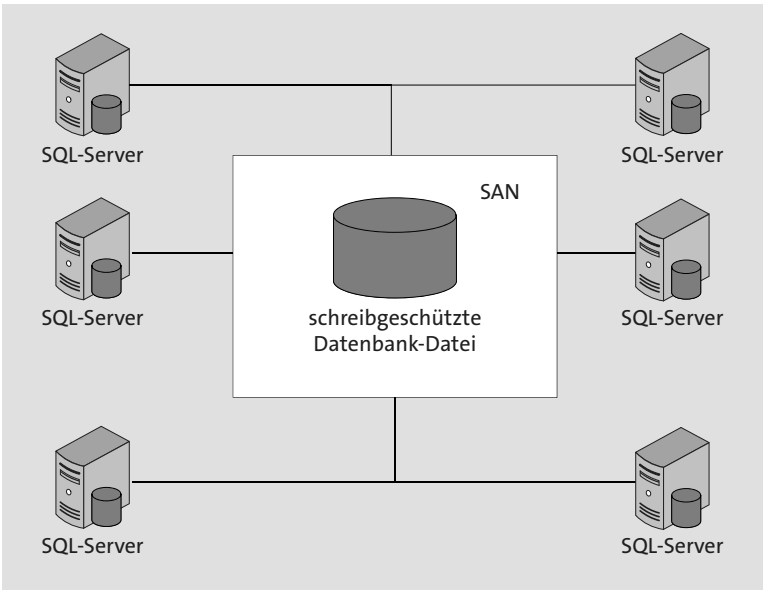

Abbildung 13.7 Skalierbare freigegebene Datenbanken

Wie Sie sehen, ist auch das eine Skalierungsmöglichkeit. SQL Server bietet diese Form der Skalierung seit der Version 2008 R2 an.

13.5 Skalierbarkeit von Datenbanken mit Hilfe der Peer-to-Peer-Transaktionsreplikation

Eine *Peer-to-Peer-Transaktionsreplikation* bietet eine weitere Möglichkeit der Skalierung. Betrachtet man die Peer-to-Peer-Replikation aus dieser Perspektive, ist sie nicht nur eine reine Lösung zum effektiven Replizieren von Daten, um deren Verfügbarkeit zu erhöhen. Sie eignet sich außerdem als effektive Skalierungslösung, so dass identische Kopien der Daten auf verschiedenen Servern verteilt liegen.

Wenn ein Knoten ausfällt, müssen auf Anwendungsebene die Schreibvorgänge für diesen Knoten auf einen anderen Knoten umgeleitet werden. Transaktionen, die Änderungen an einer Kopie der Daten vornehmen, werden über die Peer-to-Peer-Transaktionsreplikation auf alle anderen Knoten fast in Echtzeit repliziert. Davon profitieren Anwendungen, die ihre Lesevorgänge über mehrere Knoten verteilen können. Beim Schreiben verhält sich die Peer-to-Peer-Transaktionsreplikation hinsichtlich der Geschwindigkeit jedoch wie bei einem einzelnen Knoten. Der Grund dafür ist, dass jede Änderung (`INSERT`, `UPDATE`, `DELETE`) an jeden Knoten weitergegeben werden muss. Die Replikation erkennt, wenn eine Änderung auf einem bestimmten Knoten bereits ausgeführt wurde, und verhindert dessen erneute Aktualisierung. Handelt es sich um eine Anwendung, die immer die neuesten Daten wiedergeben muss, kann dies aufgrund der Verzögerung beim Schreiben problematisch sein.

Weitere Informationen zur Peer-to-Peer-Transaktionsreplikation finden Sie online bei Microsoft unter *https://docs.microsoft.com/de-de/sql/relational-databases/replication/transactional/peer-to-peer-transactional-replication?view=sql-server-ver15*.

13.6 AlwaysOn – nicht nur ein Thema für Hochverfügbarkeit

Eine der neuen Funktionen seit SQL Server 2012 in der Enterprise Edition ist *AlwaysOn*. In Kapitel 15, »Hochverfügbarkeitslösungen«, werden wir das Thema aus Sicht einer Hochverfügbarkeitslösung besprechen. In diesem Abschnitt möchten wir daher nicht auf diese Sichtweise eingehen, sondern das Thema aus der Perspektive einer Skalierung betrachten.

Es gibt unter AlwaysOn einen primären SQL Server, der die Transaktionen der Anwendungen im Unternehmen ausführt. Dazu kommen vielleicht das unternehmensweite Reporting, ein Data Warehouse und Abfragen, die ad hoc ausgeführt werden.

Nachts laufen zusätzlich ETL-Prozesse und Wartungspläne mit den Backups für die Datenbanken.

Worauf wir hinauswollen, ist die Tatsache, dass der SQL Server die gesamte Last trägt. Unter AlwaysOn kommen sekundäre Instanzen von SQL Server zum Einsatz. Die Daten der primären Instanz werden auf die sekundären Instanzen synchronisiert. Dieses Prinzip ist dem der Datenbankspiegelung ähnlich, jedoch hat es den Vorteil, dass bei AlwaysOn ein lesender Zugriff auf die sekundären Instanzen möglich ist. Wenn wir das aus der Sicht der Skalierung betrachten, ist es z. B. möglich, das Reporting oder die Daten für ein Data Warehouse auf eine sekundäre Instanz zu verlagern. Es geht noch weiter: Prozesse, die im Hintergrund lesend auf Daten zugreifen, bzw. Backups können so auf sekundäre Instanzen ausgelagert werden. Das verteilt die Last und sorgt gleichzeitig für eine Entlastung der primären Instanz.

Wenn Sie zum Thema AlwaysOn vertiefende Informationen suchen, schlagen Sie in Kapitel 15, »Hochverfügbarkeitslösungen«, nach.

13.7 Service Broker – Skalierung für Entwickler

Service Broker ist ein Framework für Entwickler, die hochskalierbare Anwendungen auf Basis des SQL Servers mit asynchroner Kommunikation entwickeln möchten. In dieser Aussage stecken die wesentlichen Vorteile von Service Broker. Er ist ein Dienst für asynchrone Kommunikation. Er ist skalierbar und als Framework ausgelegt.

Service Broker kann z. B. eingesetzt werden, um Nachrichten innerhalb einer Datenbank oder von einer Datenbank an eine andere zu versenden. Dabei ist es möglich, Anwendungen auf Basis von Service Broker zu entwickeln und später, wenn die Notwendigkeit besteht, über Instanzen hinweg zu skalieren. In diesen Fällen müssen lediglich einige Änderungen, wie z. B. geänderte Routen, vorgenommen werden, um die Anwendung zu skalieren.

Entwickler können dieses Framework nutzen, um Service-Broker-Objekte anzulegen, wie z. B. Nachrichtentypen, Warteschlangen für Nachrichten und Routen. Die meisten Service-Broker-Objekte haben Namen in einem URI-Format. Das dient der eindeutigen Identifizierung der Objekte. Bevor Sie jedoch Service Broker nutzen können, müssen Sie ihn für die Datenbank aktivieren.

```
ALTER DATABASE [AdminSchnelleinstiegDB] SET ENABLE_BROKER;
```

Listing 13.1 Aktivieren von Service Broker für die Datenbank

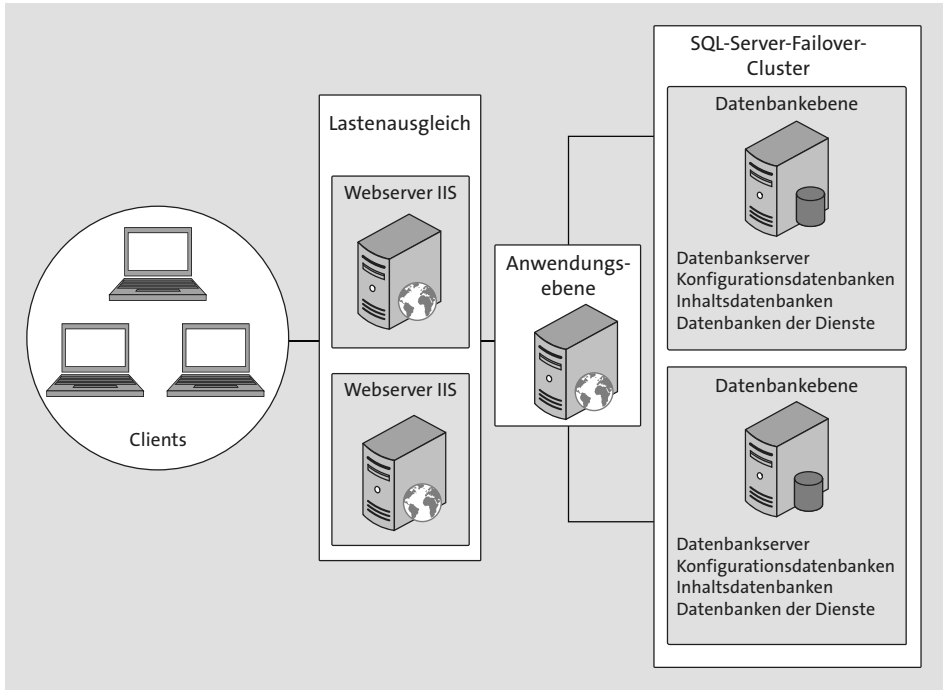

Abbildung 13.8 Übersicht Service-Broker-Architektur

13.7.1 Nachrichtentypen

Ein *Nachrichtentyp* definiert den Namen und Inhalt einer zur Versendung anstehenden Nachricht. Ein Nachrichtentyp kann leer sein – das heißt, der Textteil der Nachricht enthält keine Daten – oder aber wohlgeformtes XML (syntaktisch richtig) enthalten. Auch die Bindung an ein Schema ist möglich, was dazu beiträgt, dass der Inhalt der XML-Nachricht klaren Regeln unterliegt. Ein Schema sorgt dann für die richtigen Datentypen, die exakte Reihenfolge der Elemente und Attribute. Ein Nachrichtentyp, der nicht einer Überprüfung unterliegt, kann binäre Daten enthalten, XML oder leer sein. Einen neuen Nachrichtentyp mit wohlgeformtem XML erstellen Sie für Service Broker mit der folgenden Anweisung:

```
CREATE MESSAGE TYPE ServericeBrokerNachricht
VALIDATION = NONE
```

Listing 13.2 Festlegen des Nachrichtentyps

13.7.2 Verträge

Mit *Verträgen* werden die Nachrichtentypen für die Konversation mit Service Broker festgelegt. Außerdem bestimmt ein Vertrag die Richtung der Nachricht, das heißt,

wer der Sender ist. Ein Vertrag kann daher als eine Vereinbarung zwischen den Diensten angesehen werden.

```
CREATE CONTRACT ServiceBrokerVertrag
(ServericeBrokerNachricht SENT BY INITIATOR);
```

Listing 13.3 Erstellen des Vertrags

13.7.3 Warteschlangen

In den *Service-Broker-Warteschlangen* werden Nachrichten gespeichert. Somit ist asynchrone Kommunikation zwischen den Diensten möglich, und Nachrichten können den Warteschlangen zur weiteren Verarbeitung entnommen werden.

```
CREATE QUEUE ServiceBrokerSendeWarteschlange;
CREATE QUEUE ServiceBrokerEmpfangsWarteschlange;
```

Listing 13.4 Erstellen der Sende- und Empfangswarteschlange

13.7.4 Dienste

Dienste sind adressierbare Endpunkte für die Nachrichtenübermittlung in Service Broker. Service-Broker-Nachrichten werden von einem Dienst zu einem anderen gesendet. Jeder Dienst verfügt über eine Warteschlange zum Einreihen von Nachrichten. Es werden zwei Dienste in Service Broker erstellt, einer zum Senden und einer zum Empfangen von Nachrichten.

```
CREATE SERVICE ServiceBrokerSendeService
ON QUEUE ServiceBrokerSendeWarteschlange
(ServiceBrokerVertrag);

CREATE SERVICE ServiceBrokerEmpfangsService
ON QUEUE ServiceBrokerEmpfangsWarteschlange (ServiceBrokerVertrag);
```

Listing 13.5 Erstellt einen Service-Broker-Dienst

Nachdem Sie die Service-Broker-Objekte erstellt haben, schaffen Sie die Voraussetzungen für die Dialogkonversation. Dazu legen Sie zuerst Variablen an, eine für die Dialog-ID und eine für die Nachricht.

```
DECLARE @ServiceBrokerDialog uniqueidentifier;
DECLARE @Message NVARCHAR(128);
BEGIN DIALOG CONVERSATION @ServiceBrokerDialog
FROM SERVICE ServiceBrokerSendeService
```

```
TO SERVICE ' ServiceBrokerEmpfangsService '
ON CONTRACT ServiceBrokerVertrag
WITH ENCRYPTION = OFF;
```

Listing 13.6 Einleiten der Service-Broker-Dialogkonversation

Anschließend kann die Dialogkonversation unter der Dialog-ID beginnen, gekennzeichnet durch die Richtung und den Vertrag. Die Möglichkeit, Nachrichten zu verschlüsseln, besteht über die Angabe von Encryption. Als Nächstes erzeugen Sie Variablen für die Nachrichten:

```
SET @Message = N'Servicebroker';
SEND ON CONVERSATION @ServiceBrokerDialog
MESSAGE TYPE ServericeBrokerNachricht (@Message);

SET @Message = N'Der Nachrichtenaustausch';
SEND ON CONVERSATION @ServiceBrokerDialog
MESSAGE TYPE ServericeBrokerNachricht (@Message);

SET @Message = N'ist asynchron';
SEND ON CONVERSATION @ServiceBrokerDialog
MESSAGE TYPE ServericeBrokerNachricht (@Message);
```

Listing 13.7 Versenden der Service-Broker-Nachrichten

Nachdem die Nachrichten versendet worden sind, können sie den Warteschlangen entnommen und angezeigt werden.

```
SELECT CONVERT(NVARCHAR(MAX), message_body) AS Message
FROM ServiceBrokerEmpfangsWarteschlange
```

Listing 13.8 Anzeigen der Service-Broker-Nachricht

Im folgenden Kasten finden Sie das komplette Beispiel zu Service Broker. Möchten Sie das fertige Skript einsetzen, können Sie es direkt aus den im Bonusbereich erhältlichen Materialien ausführen.

Service Broker – Beispiel

```
CREATE DATABASE ServiceBrokerTest
GO
USE ServiceBrokerTest
go
ALTER DATABASE sqlxpert SET ENABLE_BROKER;
```



```
go
CREATE MESSAGE TYPE ServericeBrokerNachricht
VALIDATION = NONE;
go
CREATE CONTRACT ServiceBrokerVertrag
(ServericeBrokerNachricht SENT BY INITIATOR);
go
CREATE QUEUE ServiceBrokerSendeWarteschlange;
CREATE QUEUE ServiceBrokerEmpfangsWarteschlange;
go
CREATE SERVICE ServiceBrokerSendeService
ON QUEUE ServiceBrokerSendeWarteschlange
(ServiceBrokerVertrag);
CREATE SERVICE ServiceBrokerEmpfangsService
ON QUEUE ServiceBrokerEmpfangsWarteschlange
(ServiceBrokerVertrag);
GO
DECLARE @ServiceBrokerDialog uniqueidentifier
DECLARE @Message NVARCHAR(128)
BEGIN DIALOG CONVERSATION @ServiceBrokerDialog
FROM SERVICE ServiceBrokerSendeService
TO SERVICE 'ServiceBrokerEmpfangsService'
ON CONTRACT ServiceBrokerVertrag
WITH ENCRYPTION = OFF
SET @Message = N'Servicebroker';
SEND ON CONVERSATION @ServiceBrokerDialog
MESSAGE TYPE ServericeBrokerNachricht (@Message);

SET @Message = N'Der Nachrichtenaustausch';
SEND ON CONVERSATION @ServiceBrokerDialog
MESSAGE TYPE ServericeBrokerNachricht (@Message);

SET @Message = N'ist asynchron';
SEND ON CONVERSATION @ServiceBrokerDialog
MESSAGE TYPE ServericeBrokerNachricht (@Message);
GO
SELECT CONVERT(NVARCHAR(MAX), message_body) AS Message
FROM ServiceBrokerEmpfangsWarteschlange;
--Bereinigen und Löschen der Beispieldatenbank
USE master
GO
DROP DATABASE ServiceBrokerTest
GO
```