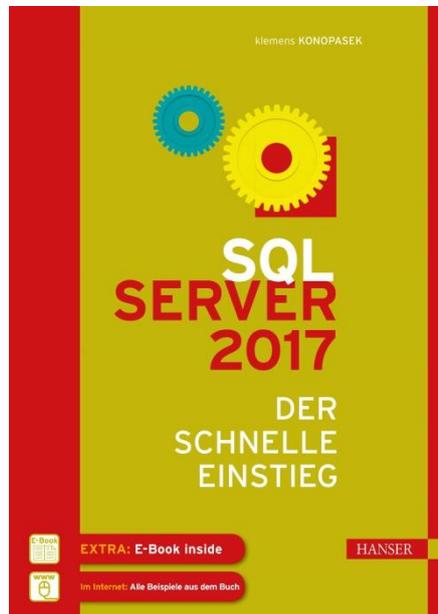


HANSER



Leseprobe

zu

SQL Server 2017

-

Der schnelle Einstieg

von Klemens Konopasek

ISBN (Buch): 978-3-446-44826-1

ISBN (E-Book): 978-3-446-44916-9

Weitere Informationen und Bestellungen unter

<http://www.hanser-fachbuch.de/>

sowie im Buchhandel

© Carl Hanser Verlag, München

Inhalt

Vorwort	XIII
1 Der SQL Server 2017 stellt sich vor	1
1.1 SQL Server – wer ist das?	2
1.1.1 Der SQL Server im Konzert der Datenbanksysteme	2
1.1.2 Entscheidungsszenarien für Datenbanksysteme	5
1.1.3 Komponenten einer Datenbankanwendung	7
1.1.4 SQL Server – das Gesamtkonzept	10
1.2 Versionen und Editionen des SQL Servers	12
1.3 SQL Server 2017 installieren	17
1.4 Datenbanken installieren und nutzen	41
1.5 Gratis: die Express Edition	47
1.6 SQL Server Feature Pack	49
2 Die grafischen Tools des SQL Server 2017	53
2.1 Die Tools im Überblick	53
2.2 Das Management Studio	56
2.3 Das Kommandozeilentool: SQLCMD	76
2.4 Der Konfigurations-Manager	78
2.5 Das SQL Server-Installationscenter	81
2.6 Der Profiler	82
2.7 Der Datenbankoptimierungsratgeber	83
2.8 Die SQL Server Data Tools	85
2.9 Der Import/Export-Assistent	89
2.10 Der SQL Server Migration Assistant	99
2.11 SQL Operations Studio	103
3 Eine neue Datenbank erstellen	105
3.1 Erstellen einer neuen Datenbank	105
3.1.1 Bestandteile einer Datenbank	105

3.1.2	Datenbank mit dem grafischen Tool anlegen	109
3.1.3	Datenbank über eine SQL-Anweisung erstellen	116
3.1.4	Datenbank mit Filestream ausstatten	117
3.2	Tabellen in der Datenbank erstellen	122
3.2.1	Tabellenfelder definieren	123
3.2.2	Spalteneigenschaften	125
3.2.3	Constraints	128
3.2.4	Indizierung	138
3.2.5	Erste Daten erfassen	146
3.3	Datenbankdiagramme einsetzen	149
3.4	Richtlinien für Benennungsregeln einsetzen	153
3.5	Was Sie noch wissen sollten	158
3.5.1	Tabellen in anderen Dateigruppen speichern	158
3.5.2	Tabellen direkt mit DDL-Anweisungen erstellen	160
3.5.3	Gefahren der grafischen Oberfläche	161
3.5.4	Berechnete Spalten integrieren	165
3.5.5	Objekte und Datenbanken skripten	168
3.6	Tabelle mit Filestream und FileTable	171
3.6.1	Tabelle mit Filestream erstellen	172
3.6.2	Objekte in einer FileTable speichern	179
3.7	Beispieldatenbank generieren	192
3.8	Speicheroptimierte Tabellen	193
3.8.1	Datenbank mit In-Memory-Filegroup erstellen	194
3.8.2	Speicheroptimierte Tabelle anlegen	196
3.8.3	Index für speicheroptimierte Tabellen	200
3.8.4	Speichernutzung beschränken	205
4	SQL – Zugriff auf Daten	207
4.1	Einsatz des Abfrage-Designers	209
4.1.1	Die Bereiche des Abfrage-Designers	209
4.1.2	Erstellen einer Abfrage	215
4.2	Sichten für den Datenzugriff gestalten	229
4.2.1	Gründe für den Einsatz von Sichten	230
4.2.2	Erstellen einer Sicht	231
4.2.3	Daten aus einer Sicht abrufen	233
4.3	SQL-Anweisungen verwenden	235
4.3.1	Data Query Language (DQL)	236
4.3.2	Data Manipulation Language (DML)	246
4.3.3	Die MERGE-Anweisung	247
4.3.4	Den Abfrage-Designer im Abfrageeditor einsetzen	252
4.4	Abfragen mit Geodaten	254
4.4.1	Typen im Geodatenmodell	255
4.4.2	Geodaten in Tabellen speichern und verwenden	261
4.4.3	Index für räumliche Daten	278

5	Transact-SQL – die Sprache zur Serverprogrammierung	283
5.1	Bestandteile und Funktionalität von Transact-SQL	285
5.1.1	Variablen und Datentypen	285
5.1.2	Benutzerdefinierte Tabellentypen	293
5.1.3	Funktionen	295
5.1.4	Kontrollstrukturen	332
5.1.5	Cursor für Datenzugriffe einsetzen	349
5.2	Transaktionen gezielt steuern	355
5.2.1	Automatische Transaktionen	356
5.2.2	Explizite und implizite Transaktionen	357
5.2.3	Benannte Transaktionen	363
5.3	SET-Optionen verwenden	364
5.4	Fehlerbehandlung in den Code einbauen	372
5.5	Sequenzen	378
5.6	Paging mit OFFSET und FETCH	381
5.7	Window-Funktionen	382
6	Gespeicherte Prozeduren, Funktionen und Trigger	385
6.1	Gespeicherte Prozeduren programmieren	386
6.1.1	Aufbau einer gespeicherten Prozedur	388
6.1.2	Erzeugen einer gespeicherten Prozedur	389
6.1.3	Einfache gespeicherte Prozeduren	399
6.1.4	Gespeicherte Prozeduren mit Eingabeparametern	402
6.1.5	Ergebnisrückgabe von Prozeduren	405
6.1.6	Cursor in gespeicherten Prozeduren nutzen	416
6.1.7	Transaktionen in Prozeduren	424
6.1.8	Table-Valued Parameter einsetzen	429
6.1.9	Systemintern kompilierte gespeicherte Prozeduren	434
6.1.10	Gespeicherte Prozeduren aus Client-Anwendungen heraus aufrufen	445
6.2	Mit Triggern automatisieren	460
6.2.1	DML-Trigger: Insert, Update, Delete	461
6.2.2	Triggerreihenfolge festlegen	481
6.2.3	INSTEAD OF-Trigger	483
6.2.4	Rekursive Trigger	486
6.2.5	Trigger löschen	498
6.2.6	Systemeigen kompilierte Trigger	499
6.2.7	DDL-Trigger	504
6.3	Benutzerdefinierte Funktionen implementieren	511
6.3.1	Skalarwertfunktionen	511
6.3.2	Inline-Funktionen	517
6.3.3	Tabellenwertfunktionen	519
6.3.4	Systemintern kompilierte benutzerdefinierte Funktionen	525

6.4	Debuggen	528
6.4.1	Voraussetzungen für das Debuggen	528
6.4.2	Debuggen einer gespeicherten Prozedur	530
6.4.3	Debuggen von Triggern	535
6.4.4	Debuggen von Funktionen	538
6.5	Praxistipps	539
6.5.1	Fehleranalyse mit ERROR_MESSAGE()	540
6.5.2	Fehler gezielt zur Ablaufsteuerung einsetzen	542
6.5.3	Fehlerprotokoll führen	545
6.5.4	Über Fehler benachrichtigen lassen	547
6.5.5	Automatisierte Importe mit BULK INSERT	551
7	SQL Server CLR-Integration	557
7.1	Mit im Boot: .NET Framework	558
7.1.1	Integration mit dem Visual Studio	560
7.2	CLR-Aktivierung	563
7.2.1	Code auf den Server bringen: Assembly	566
7.3	.NET User-Defined Functions	569
7.4	.NET Stored Procedures	577
7.4.1	Datenzugriff aus der CLR heraus	577
7.4.2	Prozeduren mit Werterückgabe	578
7.4.3	Zugriff auf externe Daten	585
7.5	.NET-Trigger	592
7.6	User-Defined Aggregates (UDA)	601
7.7	Externe Assemblys verwenden	608
7.8	CLR-Sicherheitseinstellungen	615
7.8.1	Assembly als vertrauenswürdig erklären	616
7.8.2	Assembly signieren	620
7.9	Verwalten des Servers mit SMO	628
7.10	Übrigens: Debuggen	634
7.10.1	Debuggen einer T-SQL Stored Procedure	635
7.10.2	Debuggen einer .NET-Stored Procedure	637
8	Data Tier Applications und SQL Server Data Tools	641
8.1	Datenebenenanwendungen	641
8.1.1	DAC über Management Studio erstellen	642
8.1.2	Eine DAC auf dem SQL Server bereitstellen	645
8.1.3	Aktualisieren einer DAC	647
8.1.4	Entfernen einer DAC	650
8.1.5	Von DACPAC zu BACPAC	650
8.1.6	Erstellen einer DAC mit dem Visual Studio	655
8.2	Die SQL Server Data Tools	656
8.2.1	Ein neues Datenbankprojekt erstellen	656

8.2.2	Datenbankobjekte erstellen	658
8.2.3	Datenbankprojekt bereitstellen	663
8.2.4	Schemavergleich	666
8.2.5	Datenbank in ein Datenbankprojekt importieren	671
8.2.6	Ersatz für das Management Studio?	674
9	Client-Server-Datenbank verwalten	677
9.1	Anfügen und Trennen von Datenbanken	677
9.1.1	Trennen einer Datenbank	678
9.1.2	Anfügen einer Datenbank	681
9.1.3	Option „Automatisch schließen“	687
9.2	Datenbank sichern	688
9.2.1	Sicherungsvarianten	688
9.2.2	Sicherungsziele	690
9.2.3	Sicherung mit dem Management Studio	693
9.2.4	Sicherung über TRANSACT-SQL	699
9.2.5	Zeitgesteuerte Sicherung mit dem SQL Server-Agent	703
9.2.6	Zeitgesteuerte Sicherung mit der Express Edition	707
9.2.7	Datenbank wiederherstellen	711
9.2.8	Einsatz der Zeitachse beim Wiederherstellen	715
9.2.9	Wiederherstellung über Transact-SQL	720
9.2.10	Desaster Recovery	721
9.2.11	Recovery mit FILESTREAM	728
9.3	Datenänderungen protokollieren	731
9.3.1	Change Data Capture	731
9.3.2	Temporale Tabellen	737
9.4	Mit mehreren Instanzen arbeiten	763
9.4.1	Standardinstanzen und benannte Instanzen	764
9.4.2	Zugriff auf Instanzen steuern	766
10	Sicherheit und Zugriffsberechtigungen	771
10.1	Authentifizierungsmodi - Anmeldungen und Benutzer	771
10.1.1	Windows-Authentifizierung	773
10.1.2	Gemischter Modus	774
10.1.3	Anmeldung und Benutzer	774
10.2	Berechtigungen	776
10.3	Rollen	777
10.3.1	Serverrollen	777
10.3.2	Datenbankrollen	780
10.3.3	Anwendungsrollen	781
10.4	Anmeldeinformationen (Credentials)	782
10.5	Schema	784
10.6	Verwaltung im Management Studio	786

10.6.1	Serveranmeldung hinzufügen	786
10.6.2	Schema anlegen	792
10.6.3	Datenbankbenutzer hinzufügen	792
10.6.4	Rollen in einer Datenbank anlegen	796
10.7	Berechtigungen vergeben	798
10.7.1	Berechtigungen auf Datenbankebene	798
10.7.2	Berechtigungen auf Serverebene	806
10.8	Lösungen mit T-SQL	807
10.8.1	Sicherheitsobjekte anlegen	808
10.8.2	Generische Skripte	814
10.9	Contained Databases	814
10.10	Administratorzugriff wiederherstellen	822
10.11	Indirekte Zugriffe verwalten	826
10.11.1	Datenzugriffe über Sichten	827
10.11.2	Sicherheit mit Prozeduren erhöhen	828
10.12	Sicherheit auf Zeilenebene	835
10.12.1	Bestandteile von Row Level Security (RLS)	836
10.12.2	Sicherheitsfunktion erstellen	839
10.12.3	Security Policy definieren	842
10.12.4	Ändern von Sicherheitsrichtlinien	845
10.13	Zugriff auf andere Server	848
10.13.1	SQL Server als Verbindungsserver	850
10.13.2	Verbindungsserver mit Fremdprodukten	857
10.14	Daten verschlüsseln mit Always Encrypted	861
10.14.1	Voraussetzungen für Always Encrypted	862
10.14.2	Konfiguration von Always Encrypted	863
10.14.3	Vorhandene Daten verschlüsseln	870
10.14.4	Abfragen von verschlüsselten Daten	873
10.14.5	Erstellen von Tabellen mit verschlüsselten Spalten	877
10.14.6	Einfügen von Daten mit Verschlüsselung	880
10.14.7	Treibereinsatz am Client	883
11	Erweiterte Funktionalitäten	887
11.1	Datenbank-E-Mail	887
11.1.1	Einrichten von Datenbank-E-Mail	888
11.1.2	E-Mails aus der Anwendung heraus versenden	895
11.1.3	Varianten des E-Mail-Versands	897
11.1.4	Konfiguration über Systemprozeduren	904
11.1.5	Mailbenachrichtigung für Agent-Aufträge	910
11.2	Integration Services	918
11.2.1	Datenabgleich mit IS	919
11.2.2	Pakete ausführen und auf den Server bringen	942
11.2.3	SSIS-Projekte auf den Server bringen	944

12	SQL Server 2017 auf Linux	951
12.1	Installation des SQL Servers	953
12.2	Kommandozeilentools installieren	958
12.2.1	SQLCmd mit ODBC	958
12.2.2	mssql-cli	962
12.3	Server-Agent ergänzen	966
12.4	Integration Services	967
12.5	Serverdienst starten	968
12.6	Updates installieren	969
12.7	Weitere Konfiguration	971
12.8	Windows-Authentifizierung	976
12.9	Linux auch am Client: SQL Operations Studio	984
A	Anhang	989
A.1	Die Tabellen der Datenbank WAWI	989
Index	1001

Vorwort

Eine neue SQL Server-Version ist da! Dies bedeutet einerseits viel Freude, wieder mit neuen Features Aufgabenstellungen aus der Praxis noch besser lösen zu können, und andererseits aber auch, dass ich mich wieder hinsetzen muss, um dieses Buch für diese neue Version zu schreiben. Aber das mache ich gerne für Sie!

Und da es viele spannende Neuerungen vorzustellen gibt, ist die Seitenanzahl bei dieser Neuauflage ordentlich angestiegen. Sie werden sich vielleicht fragen, ob der Untertitel *Der schnelle Einstieg* zu einem Buch passt, das eine Stärke von über tausend Seiten aufweist. Die Antwort ist: und ob! Denn selbst in unserer schnelllebigen Zeit hat das Attribut *schnell* auch noch andere Bedeutungen als *rasch* oder *kurz*. Schlägt man den Duden auf, findet man unter dem Begriff *schnell* als erstes die beiden Verwendungen *schnellstens* und *so schnell wie möglich* vor. Diese beiden passen perfekt zum Charakter des Buches. Der Microsoft SQL Server ist ein so umfangreiches Produkt, dass ein rascher oder kurzer Einstieg gar nicht möglich sein kann. Ich bin vielmehr bemüht, durch die Auswahl der Themen und die Fokussierung auf in der Praxis relevante Schwerpunkte Sie so zu unterstützen, dass Ihr Einstieg schnellstens und so schnell wie möglich, und damit verbunden auch effizient, erfolgreich und angenehm erfolgen kann.

Der SQL Server 2017 kommt ja in sehr kurzem Abstand nach dem SQL Server 2016. Daher kommen in dieser Ausgabe Neuerungen beider Versionen zum Zug. Soweit es den SQL Server 2017 betrifft, ist die wohl unangefochten größte Neuerungen die Verfügbarkeit unter Linux. Dies ist auf Entwicklungen der letzten Jahre zurückzuführen, die zuvor absolut unvorstellbar und als Paradigmenbruch gegolten haben. So hat Microsoft in den letzten Jahren eine unheimlich umfassende Öffnung zu anderen Systemen vollzogen. Ist man vor einigen Jahren schon glücklich gewesen, wenn man auf einem Apple- oder Android-Smartphone ein Word-, Excel- oder PowerPoint-Dokument irgendwie zum Lesen anzeigen konnte, sind mittlerweile die Office-Anwendungen für viel Plattformen Standard. Mit der Öffnung der Produkte für nicht Windows-Plattformen ist ein Meilenstein in der Microsoft-Geschichte gesetzt worden, bei dem die Funktionalität und der Service im Vordergrund stehen. Parallel dazu sind die Angebote in der Cloud mit Azure derart umfangreich geworden, dass damit viele Anforderungen ohne Abstriche umgesetzt werden können. Gleichzeitig hat sich die Funktionalität und Usability von Web-Oberflächen – stelle man sich an dieser Stelle Office 365 vor – dermaßen verbessert, dass hier kaum noch Abstriche gegenüber einer Windows-Anwendung gemacht werden müssen.

Ende 2016 ist Microsoft Platinum-Mitglied der Linux-Foundation geworden und bekennt sich damit offiziell zu diesem Betriebssystem. Der SQL Server für Linux ist meiner Ansicht nach bislang die Krönung dieser Mitgliedschaft und der zuvor beschriebenen Entwicklungen. Schon davor ist der SQL Server die führende Datenbankplattform unter Windows gewesen, mit der Ausweitung auf Linux wird der Einsatzbereich noch einmal ganz deutlich vergrößert. Die Form der Implementierung ist insbesondere bemerkenswert, als der SQL Server nicht einfach für Linux nachgebaut worden ist. Die codegleiche Basisengine ist über eine neue Abstraktionsschicht auch unter dem freien Betriebssystem lauffähig geworden.

Servervirtualisierung ist unabhängig vom Betriebssystem State of the Art geworden und auch der Weg in die Cloud ist für Datenbanken an der Schwelle zur breiten Anerkennung. Die Virtualisierung und die Cloud sind endgültig auch bei der Datenbank angekommen. Dies hängt auch damit zusammen, dass sich die Virtualisierungsprodukte derart weiterentwickelt haben, dass Vorbehalte speziell für Datenbankserver nicht mehr bestehen. Es gibt es keine Nachteile mehr gegenüber einem physischen Server. Damit ist mit den Datenbanken eine der letzten Virtualisierungslücken bereits geschlossen. Ausnahmslos alle SQL Server bei meinen Kunden sind längst virtualisierte Server. Anwendungen in die Cloud auszulagern verliert langsam an Schrecken und Vorbehalte verschwinden. Mit dem Inkrafttreten der europäischen Datenschutzgrundverordnung (DSGVO) im Mai 2018 ist es besonders wichtig, dass die großen Cloud-Anbieter europäische Serverstandorte anbieten und die hundertprozentige Datenhaltung in Europa garantieren können.

Mit Windows Azure SQL-Datenbank steht eine einfach zu verwendende und leistungsstarke Cloud-Plattform für den SQL Server zur Verfügung, der Unternehmen den Betrieb eines Datenbankservers in kostengünstiger und effizienter Form ermöglicht. Um Themen wie Verfügbarkeit, Hardware und Skalierbarkeit müssen Sie sich dann keine Gedanken machen. Die Themen Virtualisierung und Cloud trennen die Entscheidungen für eine neue Server-Hardware und das Update der Datenbankversion voneinander. Ist der Umstieg auf eine neue Datenbankversion in der Vergangenheit mit dem Tausch der Server-Hardware einhergegangen, kann aufgrund der beschriebenen Entwicklungen ein Umstieg wesentlich zügiger vorstattengehen. Sie müssen nicht so lange auf den Einsatz der tollen neuen Features warten.

Die Neuerungen

Die Neuerungen des SQL Server 2016/2017 gegenüber ihrer Vorversion sind im Bereich der Datenbankengine auf mehrere Schwerpunkte fokussiert. Daten unter dem Schlagwort „In-Memory OLTP“ zur Gänze im Arbeitsspeicher zu halten ist als Feature ganz enorm verbessert worden, temporale Tabellen ermöglichen das Abfragen der Daten und deren Veränderung über die Zeit und die Datensicherheit steigt mit der Verschlüsselung von Daten und der Möglichkeit, Zugriffe auf Datensebene zu steuern. Zusätzlich bekommt der SQL Server 2017 mit dem SQL Server Management Studio 17 eine neue Version, welche auch die Nutzung des SQL Server unter Linux unterstützt. Auch wenn die altbekannten Clientprogramme auf Windows beschränkt bleiben, drängen neue Werkzeuge nach, die auch unter Linux und MacOS verfügbar sind. Diese sind das Visual Studio Code, das SQL Operations Studio und das Kommandozeilentool mssql-cli.

Verbesserte Werkzeuge für die Entwicklung unterstützen die Arbeit in einheitlicher Form für alle Plattformen. Die einheitliche Entwicklungsoberfläche stellt eines der Schwerpunkt-

themen dar. Die Bereiche Datenbank- und Anwendungsentwicklung wachsen immer näher zusammen. Sehen Sie sich das an, Sie werden sicher auch begeistert sein.

Für wen ist das Buch gedacht

Dieses Buch richtet sich an all diejenigen, die sich in SQL Server 2017 einarbeiten möchten. Es sind nicht nur Einsteiger in dieses Thema und dieses Produkt, sondern auch Umsteiger von MS Access und Softwareentwickler, die Datenbankkenntnisse für die Umsetzung ihrer Projekte benötigen. Das Buch ist bemüht, aus der Vielzahl an Möglichkeiten jene Themen herauszufiltern, die für das Arbeiten mit dem Produkt besonders wichtig sind und am häufigsten in der Praxis benötigt werden. Insofern habe ich für Sie mit der Auswahl der Inhalte eine Vorentscheidung getroffen, die Ihnen durch die Konzentration auf das Wesentliche den schnellen Einstieg erleichtern soll. Mit den in diesem Buch vermittelten Kenntnissen werden Sie in die Lage versetzt, effizient und umfassend mit dem neuen SQL Server zu arbeiten. Auch Umsteiger von früheren SQL Server-Versionen werden hier wertvolle Informationen für ihre weitere Arbeit mit dem Produkt finden. Schließlich sind nicht nur neue Features hinzugekommen, auch so manche altbekannte Funktionalität ist nun an einer anderen Stelle und manchmal unter einem neuen Namen anzutreffen. Dies ist vor allem für viele, die eine oder mehrere Versionen des SQL Servers übersprungen haben, eine wertvolle Hilfe.

Unter der Systemumgebung Windows hat der SQL Server mittlerweile die absolute Marktführerschaft bei Client-Server-Datenbanken erlangt. Ein großer Vorteil ist: Um auch anspruchsvolle Anwendungen zu realisieren, kann ein und dasselbe Datenbankmodul des SQL Servers plattformübergreifend verwendet werden: angefangen bei Notebooks unter Microsoft Windows 10 bis hin zu großen Multiprozessor-Servern unter Microsoft Windows Server 2016 Datacenter Edition. Mit dem SQL Server 2017 für Linux fällt für viele ein letzter Nachteil für den SQL Server bei der Auswahl eines Datenbanksystems weg.

Aufbau des Buches

Die Abschnitte des Buches sind so aufgebaut, dass Sie direkt an Ihrem Computer arbeiten und die Anwendungen unmittelbar durch Nutzung des SQL Servers ausprobieren und realisieren können. Zum Aufbau des Buches im Einzelnen:

Im **ersten Kapitel** gebe ich Ihnen einen Einstieg in die Leistungsmerkmale und Anwendungspotenziale des SQL Server 2017. Neben der Vorstellung der Editionen sowie der Erläuterung der Vorgehensweise zur Installation erfahren Sie, welche Voraussetzungen Ihr System für den Einsatz von SQL Server 2017 erfüllen muss.

Im **zweiten Kapitel** lernen Sie die Tools kennen, mit denen Sie auf den SQL Server zugreifen können. Sie benötigen diese, um den SQL Server zu verwalten und auf ihm Datenbanken zu erstellen, aber auch um mit ihm Anwendungen optimal entwickeln zu können. Hier kommen Sie erstmals mit dem SQL Server Management Studio in Kontakt, welches das wichtigste dieser Tools ist und sowohl für die Programmierung als auch die Administration eingesetzt wird.

Das **dritte Kapitel** befasst sich mit der Erstellung einer Datenbank, dem Anlegen von Tabellen und dem Einrichten von Beziehungen. Sie erfahren dabei, aus welchen Komponenten eine SQL Server-Datenbank besteht, und lernen gleichzeitig, Datenintegrität durch den Einsatz von Constraints zu implementieren. Der Einsatz von Datenbankdiagrammen, die nicht

nur zum Erstellen von Tabellen und Beziehungen dienen, sondern auch ein ideales Tool zur Dokumentation einer Datenbank sind, wird ebenso beschrieben. Die FileTables kommen in diesem Kapitel auch nicht zu kurz. Kopieren Sie Dateien in einen Ordner auf einem Netzwerk-Share, und schon tauchen diese automatisch wie von Geisterhand in der Datenbank auf.

Im Regelfall wollen Sie nicht ausschließlich Daten in eine Datenbank einpflegen, sondern natürlich Informationen auch wieder aus dem System entnehmen. Zu diesem Zweck erfahren Sie im **vierten Kapitel**, wie Sie effizient durch den Einsatz von Abfragen, Sichten und SQL-Anweisungen auf Daten zugreifen. Sie erhalten dabei auch einen kompakten Überblick über die wichtigen Sprachbereiche und Anweisungen von SQL (Structured Query Language).

Kapitel 5 bietet Ihnen einen Überblick über die Datenbanksprache Transact-SQL, die Ihnen sowohl bei der Datenbankprogrammierung als auch bei der Verwaltung von Datenbanken wertvolle Dienste leistet. So können alle Aufgaben, die Sie mit einem grafischen Verwaltungstool erledigen, auch direkt über diese Sprache realisiert werden. Dadurch können Sie solche Aufgaben in Ihre Applikationen einbauen oder sich Ihre eigenen Verwaltungstools zusammenstellen. Dieses Kapitel erläutert Ihnen die Sprachkomponenten und die dabei verwendeten Strukturen. In der Übersicht der wichtigsten Funktion finden Sie auch jene, die beim SQL Server 2017 neu hinzugekommen sind.

Nach der allgemeinen Einführung in Transact-SQL lesen Sie in **Kapitel 6**, wie Sie diese Sprache zur Programmierung von gespeicherten Prozeduren (Stored Procedures) einsetzen. Durch den gezielten Einsatz solcher Prozeduren bilden Sie die datenbezogenen Vorgänge Ihrer Datenbankapplikation auf dem Server ab. Diese müssen dann von den verschiedenen Client-Programmen nur noch aufgerufen werden. So realisieren Sie effiziente Client-Server-Applikationen.

Transact-SQL wird aber auch zur Programmierung von Triggern verwendet, die es Ihnen erlauben, Automatismen in Ihre Datenbank zu integrieren, die auf das Einfügen, Ändern und Löschen von Datensätzen reagieren. Besonders interessant für die Praxis sind mittlerweile auch Datenbanktrigger, mit denen Sie sowohl Änderungen an der Datenbankstruktur überwachen als auch bei Bedarf unterbinden können. Des Weiteren lernen Sie die benutzerdefinierten Funktionen (User-Defined Functions, UDFs) kennen. Diese Funktionen können im Gegensatz zu gespeicherten Prozeduren auch in SQL-Anweisungen eingesetzt werden und erweitern dadurch den Einsatzbereich in der Programmierung von Transact-SQL. Sie können sie darüber hinaus auch verwenden, um die Standardfunktionen vom SQL Server zu erweitern.

Das **Kapitel 7** beschäftigt sich mit dem Thema .NET im Zusammenhang mit dem SQL Server. Sie lesen hier nicht nur, wie Sie Prozeduren, Funktionen und Trigger mit einer .NET-Programmiersprache für die SQL Server CLR (Common Language Runtime) entwickeln, sondern auch, wie Sie Aggregatfunktionen selbst programmieren. Diese stehen Ihnen dann innerhalb von SQL-Anweisungen wie andere Aggregatfunktionen zur Verfügung. Ein wesentliches Augenmerk habe ich dabei auf die neuen Sicherheitsanforderungen, die für das Ausführen von CLR-Code ab dem SQL Server 2017 erfüllt sein müssen, gelegt. Lesen Sie dazu in diesem Kapitel, wie Sie Ihren Programmcode mit Zertifikaten signieren.

Die Server Management Objects (SMO), mit denen Sie auf so gut wie alle Funktionalitäten des SQL Servers programmatischen Zugriff haben, runden das Kapitel ab. Durch die SQL Server Data Tools wird die Programmierung für die SQL Server CLR interessant, da dazu ein extrem leistungsstarkes und dazu noch freies Werkzeug verwendet werden kann.

Die SQL Server Data Tools revolutionieren für Entwickler die Arbeit mit der Datenbank. Daher sind sie es mir wert, gemeinsam mit den Datenebenenanwendungen in **Kapitel 8** behandelt zu werden. Datenebenenanwendungen, oder Data Tier Applications, wie sie im Original genannt werden, sind mittlerweile schon fast schon integraler Bestandteil für viele Phasen der Datenbankentwicklung. Sie sind das Werkzeug, um Datenbanken auszurollen und Aktualisierungen und Versionierung zu organisieren. Sie sind in die SQL Server Data Tools fest integriert. Die Data Tools sind ein Werkzeug, mit dem es für Programmierer möglich ist, unter dem Dach des Visual Studios mit einem Werkzeug alle Entwicklungsaufgaben von der Datenbank bis zum Frontend zu erledigen.

Da Sie von einer Datenbank nicht viel haben, wenn Ihre wertvollen Daten nicht sicher sind, erfahren Sie in **Kapitel 9**, wie Sie eine SQL Server-Datenbank regelmäßig sichern und im Ernstfall auch wiederherstellen können. Datenbanksicherungen haben ihre Bedeutung aber nicht nur in einem Störfall, sondern sind auch in der täglichen Arbeit mit der Datenbank wichtig, weil sie zum Beispiel auch dafür verwendet werden, eine Datenbank von einem Server auf einen anderen zu übertragen. Besonders begeistert bin ich von den temporalen Tabellen, die ohne Programmieraufwand Veränderungen der Daten über die Zeit verfügbar machen.

In **Kapitel 10** finden Sie alle Informationen, die Sie für die Herstellung der Sicherheit Ihrer Datenbank benötigen. Sie lesen in diesem Kapitel, wie Sie auf Ihrem SQL Server Benutzer anlegen und diesen verschiedene Berechtigungen zuweisen. Sie erfahren, wie Sie Contained Databases einsetzen und nutzen können. Lesen Sie in diesem Kapitel auch, wie Sie den Zugriff auf Zeilenebene einschränken und Ihre Daten mit Always Encrypted verschlüsseln können. Damit können Sie den Anforderungen der europäischen Datenschutzgrundverordnung (DSGVO) mit dem SQL Server noch besser gerecht werden.

In **Kapitel 11** erläutere ich Ihnen zwei erweiterte Funktionalitäten, die Ihnen ergänzend zur Verfügung stehen, falls Sie nicht die Gratis-Edition des SQL Servers 2017 verwenden. Ich stelle Ihnen hierbei Datenbank-E-Mail sowie die Integration Services etwas genauer vor.

In **Kapitel 12** zeige ich Ihnen am Beispiel von Ubutu, wie Sie einen SQL Server unter Linux installieren und einsetzen.

Mit diesem Buch lernen Sie anhand von problembezogenen Aufgabenstellungen in anschaulicher und systematischer Form die zahlreichen Möglichkeiten des SQL Server 2017 für die Datenbankentwicklung kennen. Das Buch eignet sich sowohl zum Selbststudium als auch als begleitende Unterlage für Schulungen.



www.downloads.hanser.de

Hier finden Sie sämtliche Dateien aller im Buch verwendeten Beispiele. Diese enthalten u. a. die Beispiel-Datenbanken, SQL-Skripte zu jedem Kapitel sowie Visual Studio-Projekte.

Ich möchte mich an dieser Stelle bei meinem Dreimäderlhaus – Petra, Alina und Lea – für ihre immense Geduld bedanken.

Und nun viel Erfolg beim schnellen Einstieg in die Arbeit mit dem SQL Server 2017.

Klemens Konopasek, Gössendorf/Graz

1

Der SQL Server 2017 stellt sich vor

Der SQL Server 2017 ist da – wieder eine tolle neue Version. Auch wenn diese Version sehr rasch nach dem SQL Server 2016 erschienen ist, der als Hauptrelease bezeichnet wird, ist der SQL Server 2017 als vermeintliche Zwischenversion alles andere als unbedeutend. Während der Entwicklung als SQL Server vNext bezeichnet, ist es doch erstmals in der Geschichte des MS SQL Server soweit, dass dieser neben Windows auch für Linux als Betriebssystem verfügbar ist. Und das ist eine echte Revolution. Da diese Versionen so rasch aufeinanderfolgen, gehe ich in dieser Auflage auch auf Features ein, die mit dem SQL Server 2016 eingeführt worden sind.

Ich bin von den Neuerungen der Versionen 2016 und 2017 begeistert und ertappe mich immer wieder, wenn ich bei einem Kunden noch eine der Vorversionen vorfinde, bei dem Gedanken: „Oje, jetzt muss ich wieder auf dieses und jenes verzichten.“ Mein persönliches Highlight ist die stark verbesserte Möglichkeit, Tabellen einer Datenbank im Arbeitsspeicher zu halten. Sehr spannend finde ich auch die neuen Sicherheitsfunktionen. Wird die Version 2016 von Microsoft selbst doch als das „Performance und Sicherheits-Release“ bezeichnet. Und nun ist dies alles mit dem SQL Server 2017 auch für Linux verfügbar. Viele vermeintliche kleinere Features sind aber groß in der Auswirkung, so ist für Entwickler die native Unterstützung von JSON von großer Bedeutung.

Ich hoffe, auch Sie gehen mit Freude an den SQL Server 2017 und an dieses Buch heran!

Im ersten Kapitel möchte ich Ihnen einen Überblick über das Produkt und seine Komponenten geben. Anschließend stelle ich Ihnen die Editionen vor, in denen der SQL Server 2017 verfügbar ist, und zeige Ihnen, wie Sie bei der Installation vorgehen. Darüber hinaus werden Sie erfahren, wie Sie mit dem SQL Server 2017 arbeiten, um vorhandene Datenbanken und darin enthaltene Datenbankobjekte zu nutzen. Ebenso zeige ich Ihnen, wie eine Integration zu Client-Umgebungen erfolgen kann. Den Abschluss dieses Kapitels bilden die Besonderheiten der freien Express-Version.

■ 1.1 SQL Server – wer ist das?

Eigentlich wollte ich diesen Abschnitt ursprünglich mit „SQL Server – was ist das?“ betiteln. Aber das kam mir dann so plump vor, dass ich das „was“ durch ein „wer“ ersetzt habe. Dies klingt besser, auch wenn ich den SQL Server dadurch nicht personifizieren will.

1.1.1 Der SQL Server im Konzert der Datenbanksysteme

Wenn wir heutzutage von einer Datenbank sprechen, meinen wir in der Regel – ohne dies explizit zu erwähnen – eine relationale Datenbank. Andere Datenbanksysteme, wie zum Beispiel objektorientierte Datenbanken, konnten sich nie wirklich auf breiter Front durchsetzen oder haben ihre beste Zeit bereits hinter sich. Neue moderne Ansätze, die sich unter dem Begriff NoSQL finden, sind für sehr spezielle Anwendungsbereiche ausgerichtet und zielen darauf ab, relationale Datenbanken zu verdrängen oder gar zu ersetzen. Vielmehr wollen sie eine Ergänzung in Nischenbereichen sein, für die relationale Strukturen nicht die ideale Form sind. Daher steht das *No* auch nicht für *kein*, sondern für *not only*. Vielfach kommen diese auch aufs Tapet, wenn von Big Data die Rede ist. Microsoft trägt dem durch die Möglichkeit von hybriden Lösungen mit Hadoop oder R Server Rechnung.



HINWEIS: Relationale Datenbanksysteme entwickeln sich in die Richtung weiter, dass wichtige Features aus dem nicht-relationalen Bereich als Zusätze in die Produkte integriert werden. So bietet der SQL Server 2017 als neues Feature erstmals die Unterstützung von Graph-Daten, eine wichtige Datenart bei NoSQL-Systemen.

Beim „schnellen Einstieg in den SQL Server“ gehen wir von relationalen Datenbanksystemen aus und unterteilen diese in

- *Desktop-Datenbanksysteme* und
- *Server-Datenbanksysteme*.

Eine Datenbankanwendung besteht aus drei Komponenten:

- *Data Layer:* Der Data Layer hat die Aufgabe, Daten zu verwalten und zu speichern. Hier werden außerdem die Strukturen der Datenspeicherung definiert. Diese Aufgabe wird von der Datenbank-Engine wahrgenommen.
- *Program Layer:* Im Program Layer werden die Logiken und Abläufe des Datenzugriffs abgebildet. Hier kommen unterschiedliche Entwicklungsumgebungen zum Einsatz.
- *Presentation Layer:* Aufgabe des Presentation Layers ist es, Ausgaben aus der Datenbank darzustellen. Hierzu gehören insbesondere Benutzeroberflächen und Frontend-Komponenten, mit denen der Benutzer interagiert.

Das Hauptmerkmal eines Desktop-Datenbanksystems besteht darin, dass alle drei Komponenten auf dem Desktop anzutreffen sind. Insbesondere läuft auch die Datenbank-Engine auf dem Desktop. Werden Datenbanken eines desktopbasierten Systems auf dem Server

abgelegt, wird vom Server lediglich der File-Service genutzt, um die Daten remote zur Verfügung zu stellen.

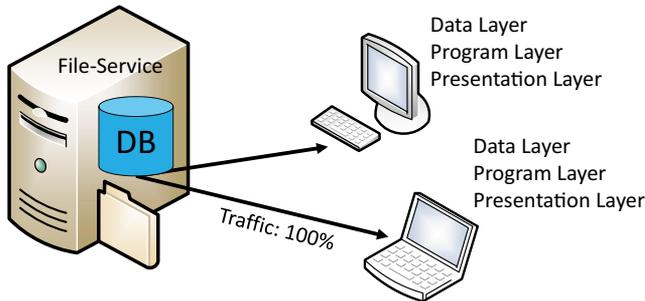


Bild 1.1 Konzept von Desktop-Datenbanken

Ein wesentliches Merkmal eines desktopbasierten Datenbanksystems ist, dass alle datenbankrelevanten Vorgänge auf dem Client ablaufen. Dazu müssen alle Daten vom Server auf den Client transferiert werden, damit die Daten von der lokalen Datenbank-Engine verarbeitet werden können.

Server-Datenbanksysteme hingegen verwenden eine Datenbank-Engine auf dem Server. Von den Clients werden Anfragen an diesen Dienst gestellt, die auf dem Server verarbeitet werden. Dadurch werden nicht alle Rohdaten, sondern nur die Ergebnisse der Anfrage an den Client gesendet. Es findet sozusagen eine Spezialisierung der Aufgaben der Datenverwaltung auf dem Server statt.

In der Abbildung ist der *Program Layer* beiden Komponenten zugeordnet, da Elemente von diesem auch in beiden Komponenten auftreten können. Wir werden später in diesem Buch zwischen serverseitiger und clientseitiger Datenbankprogrammierung unterscheiden.

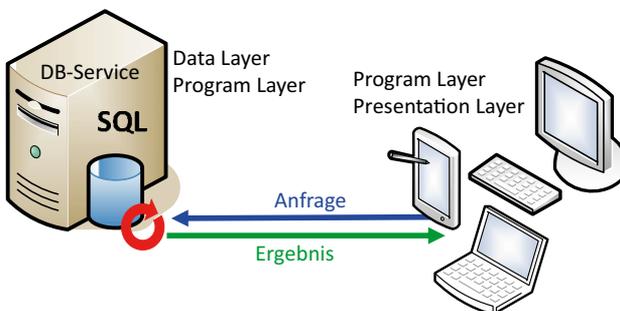


Bild 1.2 Konzept von Server-Datenbanksystemen

In der Kategorie der Desktop-Datenbanksysteme ist vor allem Microsoft Access weit verbreitet. Der SQL Server ist auch als Desktop-Datenbanksystem mit der *LocalDB* genannten Edition vertreten. Diese kann in lokal installierte Anwendungen integriert und weitergegeben werden und ist daher vor allem bei Visual Studio-Entwicklern bekannt. Ebenso in diese Kategorie einzuordnen ist SQLite, die als Embedded-DB ebenso in unzählige Anwendungen lokal integriert ist.

In der Kategorie der Server-Datenbanksysteme sind neben dem Microsoft SQL Server vor allem folgende Produkte von Bedeutung:

- Oracle
- DB2 von IBM
- SAP ASE (früher Adaptive Server Enterprise von Sybase, noch früher Sybase SQL Server)

Als Open-Source-Datenbanksysteme sind zusätzlich von Bedeutung:

- PostgreSQL
- MySQL/MariaDB

Der SQL Server ist das führende serverbasierte Datenbanksystem auf Windows-Plattformen. Bisher sind ja nur die anderen genannten Systeme auch für diverse andere Plattformen verfügbar gewesen.



HINWEIS: Der SQL Server 2017 ist erstmals für Linux verfügbar. Dies läutet einerseits eine neue Ära für den SQL Server ein, andererseits ist das die logische Konsequenz der generell von Microsoft vollzogenen Öffnung der letzten Jahre zu anderen Systemen. Wie weit der SQL Server auch auf diesen Plattformen eine bedeutende Stellung erhalten wird, wird die Zukunft zeigen.

Informationen über den SQL Server 2017 für Linux finden Sie in Kapitel 12.

ACID – das Konsistenzmodell relationaler Datenbanken

Relationale Datenbanken verwenden das Konsistenzmodell ACID. Bei diesem Modell steht die Datenkonsistenz absolut im Vordergrund und ist somit die oberste Maxime. Wenn wir uns die vier Säulen dieses Modells ansehen, werden wir feststellen, dass die Forderungen dieses Modells bei relationalen Desktop-Datenbanken wie Microsoft Access allerdings nicht erfüllt sind. Bei serverbasierten Datenbanken wie dem Microsoft SQL Server sind sie natürlich erfüllt. Die vier Säulen dieses Konsistenzmodells zeigt Bild 1.3.

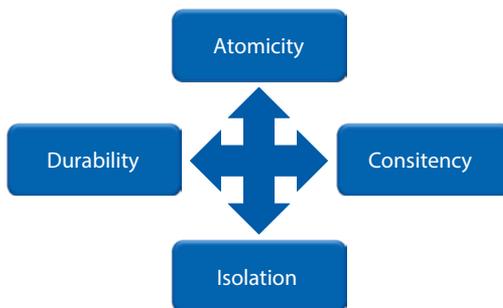


Bild 1.3 Das ACID-Konsistenzmodell

Was bedeuten diese Begriffe im Einzelnen und durch welche Mechanismen werden sie umgesetzt?

- **A – Atomicity:** Zusammenhängende Vorgänge werden entweder zur Gänze oder gar nicht durchgeführt. Gehören mehrere Schreibzugriffe zu einem gemeinsamen Vorgang, werden alle Änderungen erst übernommen, wenn auch der letzte Teilschritt erfolgreich abgeschlossen worden ist. Ist dies aus welchem Grund auch immer nicht möglich, müssen alle bisher vorgenommenen Schritte vollständig wieder rückgängig gemacht werden. Das Werkzeug, um diese Vorgabe zu erreichen, sind *Transaktionen*.
- **C – Consistency:** Die Vorgabe der Konsistenzhaltung legt fest, dass der Übergang von einem konsistenten Zustand immer nur in einen anderen konsistenten Zustand erfolgen darf. Daten müssen also immer in einem vollständigen Zustand vorliegen, es darf nie Verweise auf nicht vorhandene Daten geben. Die *Referenzielle Integrität* sorgt dafür, dass dieses Ziel erreicht wird.
- **I – Isolation:** Die Forderung der Isolation besagt, dass alle Vorgänge von anderen unbeeinflusst abgegrenzt ablaufen dürfen. Die gleichen Daten können nie zeitgleich von mehreren Personen oder Prozessen geändert werden. Solange Änderungen nicht abgeschlossen sind, sind die betroffenen Daten zumindest für den Schreibzugriff für andere gesperrt. Die Änderungen sind für den Durchführenden sofort sichtbar, für alle anderen erst nach Abschluss des Vorgangs. Auch dafür sind *Transaktionen* zuständig.
- **D – Durability:** Unter der Dauerhaftigkeit versteht man, dass Daten, die einmal festgeschrieben worden sind, dauerhaft verfügbar sind und auch Strom- und andere Systemausfälle überstehen. Dieses Ziel kann durch den Einsatz von *Protokollierung* erfolgen.

Die erwähnten Mechanismen Transaktion, referenzielle Integrität und Protokollierung werden Sie in den entsprechenden Kapiteln dieses Buches im Detail erläutert finden. Besonders interessant ist ACID im Zusammenhang mit den speicheroptimierten Tabellen, die im RAM des Servers gehalten werden. Auf den ersten Blick würde man vermuten, dass diese vor allem im Hinblick auf die Dauerhaftigkeit problematisch sind. Allerdings werden Sie lesen, dass diese Tabellen über eine entsprechende Option bei der Erstellung auch ACID-konform eingesetzt werden können. Dies wird durch das zusätzliche Ablegen der Daten auf den Disks erzielt.

1.1.2 Entscheidungsszenarien für Datenbanksysteme

Wenn Sie vor der Entscheidung stehen, ein Datenbanksystem auszuwählen, gilt es, verschiedene Gesichtspunkte zu berücksichtigen. Ich möchte Ihnen in einem kurzen Überblick die aus meiner Sicht wichtigsten Entscheidungsgründe nennen.

- *Preis (TCO):* Bei der Betrachtung der Kosten werden häufig fälschlicherweise lediglich die direkten Lizenzkosten angesetzt. Wesentlich zielführender wäre es allerdings, den Ansatz *TCO (Total Cost of Ownership)* zu wählen; denn neben den Lizenzkosten fallen zum Beispiel auch die folgenden Kosten an:
 - Kosten für Hardware
 - Kosten für Schulungen. Hierbei ist auch die Anzahl der zu schulenden Personen zu berücksichtigen. Sollen viele Personen mit einem System umgehen können oder sollen es Spezialisten für Sie erledigen?

- Kosten aufgrund von Ineffizienz, da Personen, ohne entsprechend geschult zu sein, sich statt mit ihrer eigentlichen Arbeit mit Lösungen im Desktopbereich beschäftigen.

Man kann hier keine generelle Empfehlung für ein desktop- oder serverbasiertes System aussprechen. Dies muss in der speziellen Situation beurteilt und entschieden werden.

- **Datenmenge:** Serverbasierte Systeme sind in der Lage, wesentlich größere Datenmengen zu speichern und effizienter zu verwalten als desktopbasierte Systeme.
- **Benutzeranzahl:** Nicht nur die theoretische Benutzeranzahl ist bei Serversystemen höher. Können bei Access beispielsweise theoretisch 255 Benutzer gleichzeitig auf eine Datenbank zugreifen, würde ich die tatsächliche Grenze mit 20 bis 30 gleichzeitig angemeldeten Benutzern schon als hoch angesetzt sehen. Dies ist aus der Topologie leicht zu erklären. Stellen Sie sich vor, in einem Lokal würden sich alle Kellner um einen Zapfhahn scharen und versuchen, Bier zu zapfen. Das entspricht der Logik eines Desktopsystems. Wesentlich effizienter wäre es, nur eine Person an den Zapfhahn zu stellen, die Bestellungen bearbeitet und die gezapften Biere dann an alle Kellner verteilt. Dies würde ungefähr einem serverbasierten Datenbanksystem entsprechen. Wahrscheinlich werden bei der zweiten Variante mehr Biere in der gleichen Zeit in durstigen Kehlen landen. Daher sehe ich hier klare Vorteile für ein serverbasiertes System.
- **Portabilität:** Eine Desktop-Datenbank, die oft aus einer einzigen Datei besteht, kann sehr leicht beispielsweise auf ein Notebook transferiert werden. Dies funktioniert bei einem serverbasierten System nicht so ohne Weiteres. Ersetzt man allerdings den Begriff Portabilität durch *Zugriff von überall*, könnte man darunter verstehen, auf eine Datenbank remote über eine Webapplikation zuzugreifen. Dafür wäre wiederum eine Serverdatenbank besser geeignet.
- **Flexibilität:** Eine besondere Stärke eines Desktop-Datenbanksystems liegt in der Flexibilität und Einfachheit der Anwendung. Daher wird es gerne verwendet für:
 - Auswertungen (zum Beispiel werden häufig von großen Server-Datenbanksystemen Daten importiert und danach in einem Desktop-Datenbanksystem ausgewertet),
 - Prototyping oder
 - Klein- und Kleinstlösungen.
- **Transaktionen:** Transaktionen sind für konsistente Daten unerlässlich. In der Regel werden diese nur von serverbasierten Systemen geboten.
- **Sicherheit:** Sicherheit ist unter zwei Gesichtspunkten zu betrachten.
 - Die *Zugriffssicherheit* legt fest, wer mit welchen Daten was tun darf.
 - Die *Datensicherheit* legt fest, wie sicher Daten vor Verlust geschützt sind.In beiden Bereichen liegen die Vorteile ganz klar und eindeutig bei Server-Datenbanksystemen, die hierzu spezielle Features anbieten.
- **Backup und Recovery:** Server-Datenbanksysteme ermöglichen Sicherungen im Vollbetrieb und häufig auch das verlustfreie Wiederherstellen exakt bis zum Zustand vor einem Crash. Dies gilt nicht für eine Desktop-Datenbank, bei der diese zunächst alle Anwender verlassen müssen.
- **Netzlaster:** Aufgrund der Topologie, dass nur das Ergebnis einer Anfrage vom Server an den Client übertragen wird, der diese Daten dann anzeigt und verarbeitet, können serverba-

sierte Systeme auch über schwächere Leitungen performant betrieben werden. Eine vorgegebene Bandbreite erlaubt eine größere Anzahl an Benutzern.

- *Stabilität und Verfügbarkeit:* Serversysteme verfügen über Mechanismen, welche die Verfügbarkeit der Datenbank nach dem Prinzip 24-7-365 (24 Stunden am Tag, 7 Tage die Woche und 365 Tage im Jahr verfügbar) ermöglichen.
- *Skalierbarkeit:* Durch den Einsatz unterschiedlicher Editionen ermöglichen Server-Datenbanken ein stufenloses Skalieren einer Lösung von einer kleinen Abteilungslösung bis hin zu Konzernlösungen.

Analysieren Sie Ihre Anforderungen an ein Datenbanksystem anhand dieser Kriterien und treffen Sie dann Ihre Entscheidung.



HINWEIS: Der Microsoft SQL Server bietet ein professionelles Server-Datenbanksystem zu einem vergleichsweise günstigen Preis. Mit den Editionen von Express bis Enterprise werden alle Bedürfnisse bedient; daneben erlauben sie ein uneingeschränktes Wachsen der Datenbank. Bereits ab der Express Edition können Sie die Vorteile von Sicherheit, Stabilität, Transaktionen und geringer Netzlast nutzen. Zudem ist Microsoft SQL Server ein Tool, das einfach und flexibel in der Handhabung ist wie kaum ein vergleichbares System. Außerdem sind in den letzten Versionen immer mehr Features, die nur in lizenzierten Editionen verfügbar gewesen sind, auch in die Express Edition integriert worden. Neue Features wie temporale Tabellen sind beispielsweise von Beginn an auch in der Express Edition verfügbar.

Wenn Sie sich nicht mit der Konfiguration und Wartung des SQL Servers selber auseinandersetzen möchten, können Sie den SQL Server auch in der Cloud mit Azure DB nutzen. Achten Sie aber aufgrund der Europäischen Datenschutzgrundverordnung (DSGVO) darauf, dass Ihre Daten dabei stets ausschließlich in einem Rechenzentrum innerhalb der EU gehostet werden.

1.1.3 Komponenten einer Datenbankanwendung

In der Praxis benötigen Sie keine Datenbank, sondern eine Datenbankanwendung. Auch wenn die Datenbank als „Motor“ einer Anwendung oft die wichtigste Komponente darstellt, ist ein Motor ohne ein Chassis oft nur wenig von Nutzen. Das Chassis ist die Anwendung, die aus einer Datenbank eine Datenbankanwendung macht. Eine Anwendung wird mit einer Entwicklungsumgebung erstellt und greift über standardisierte Schnittstellen mithilfe von SQL auf ein Datenbanksystem zu. Einen Überblick über einsetzbare Programmiersprachen und Schnittstellen zeigt Bild 1.4.

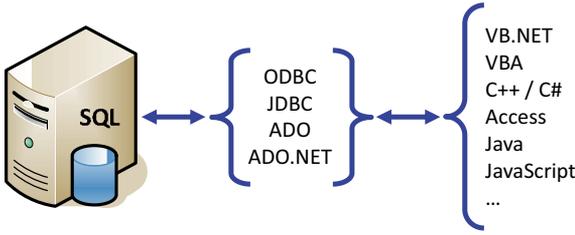


Bild 1.4 Zugriff auf eine Datenbank über Standardschnittstellen

Eine Datenbankanwendung besteht in der Regel aus folgenden Komponenten:

- Datenbankmanagementsystem als Backend für die Verwaltung der Daten
- User-Interface als Frontend für die Bedienung der Anwendung
- Server- und/oder clientseitige Programmierung für die Abbildung von Logiken

Bild 1.5 zeigt eine schematische Darstellung der einzelnen Komponenten und ihr Zusammenspiel.

HINWEIS: Der SQL Server übernimmt in diesem Szenario die Rolle des Datenbankmanagementsystems, auf das mithilfe der Abfragesprache SQL über standardisierte Schnittstellen zugegriffen wird. Für performante Lösungen ergänzt serverseitige Programmierung mittels Transact-SQL und .NET die Datenbankentwicklung mit dem SQL Server.

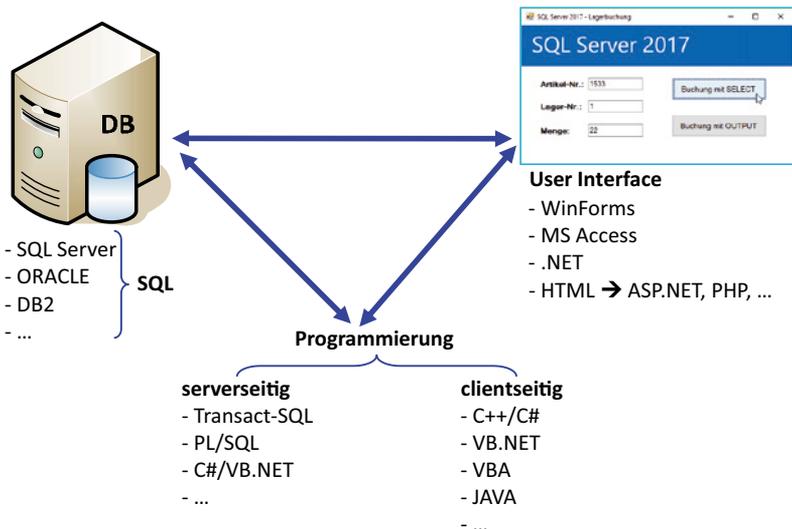


Bild 1.5 Datenbankanwendung und ihre Bestandteile

So lernen Sie den SQL Server in diesem Buch kennen:

- Den SQL Server installieren und konfigurieren
- Datenbanken und Datenbankobjekte mit dem SQL Server erstellen
- Den Zugriff auf Daten mit der Structured Query Language (SQL) vollziehen
- Serverseitig mit Transact-SQL und .NET programmieren
- Die Benutzerverwaltung zur Vergabe von Berechtigungen nutzen
- Sicherung und Wiederherstellung von Datenbanken durchführen
- Erweiterte Features einsetzen

Programmierung im Frontend und Backend

In einer Datenbankanwendung kann sowohl eine Programmierung im Frontend als auch im Backend erfolgen. Im Frontend müssen sämtliche Vorgänge im Zusammenhang mit der Benutzerführung programmiert werden. Manche Vorgänge können aber wahlweise im Frontend oder im Backend programmiert werden. Dies sind vor allem Vorgänge mit Datenbezug.

Die beiden nachfolgenden Abbildungen zeigen die Unterschiede beim Programmablauf von Programmcode, der auf dem Client oder auf dem Server läuft.

Bei clientseitiger Programmierung ist die gesamte Programmlogik im Frontend untergebracht. Werden im Ablauf Informationen aus der Datenbank benötigt oder sind Daten in die Datenbank zu schreiben, werden SQL-Anweisungen zum Datenbankserver geschickt. Mit den Ergebnissen dieser Anweisungen arbeitet der Programmcode anschließend weiter. Ein Programmablauf kann oft aus sehr vielen Einzelschritten bestehen, bei denen mitunter auch sehr viele Datenzugriffe nötig sind.

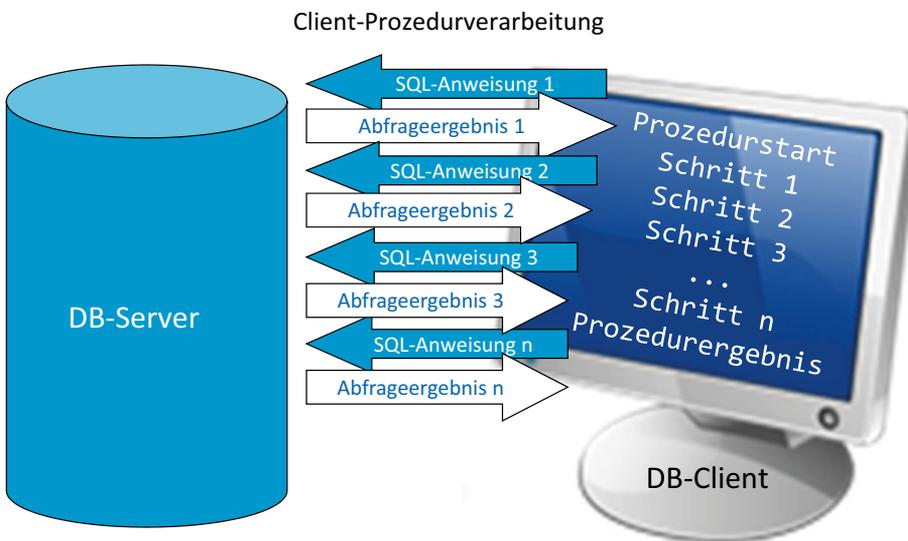


Bild 1.6 Programmlogik im Frontend

Bei serverseitiger Programmierung wird die Programmlogik beispielsweise mithilfe gespeicherter Prozeduren (Stored Procedures) im Backend umgesetzt. Der Vorteil besteht darin, dass das „Hin und Her“ zwischen Frontend und Backend entfällt. Im Frontend wird lediglich die am Server hinterlegte Funktionalität aufgerufen und das Ergebnis abgearbeitet.

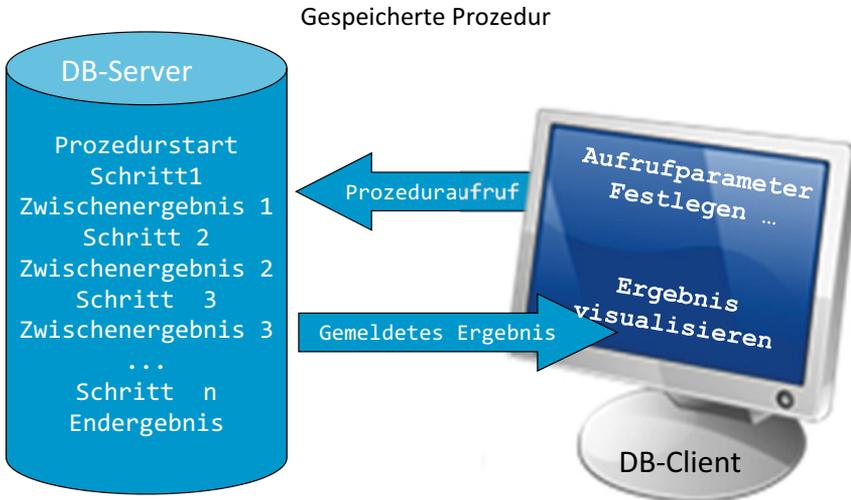


Bild 1.7 Serverseitige Programmierung

In den Kapiteln 4 bis 6 wird das Thema „Serverseitige Datenbankprogrammierung“ im Detail behandelt und auf Vor- sowie Nachteile eingegangen. Clientseitige Programmierung ist nicht Thema dieses Buches, da sie nicht vom SQL Server, sondern von der eingesetzten Programmiersprache und Entwicklungsumgebung abhängt. Anhand praktischer Beispiele, die zeigen, wie Programmierelemente des Servers von clientseitigem Code aufgerufen werden, streifen wir jedoch die clientseitige Programmierung.

1.1.4 SQL Server – das Gesamtkonzept

Der SQL Server beschränkt sich keinesfalls auf die Datenbank-Engine. SQL Server ist mittlerweile eine komplette Produktfamilie, die sich um den Kern schart. Damit ist der SQL Server nicht nur ein reines Datenbanksystem. Er bietet auch Lösungen für viele Anwendungen im Datenbanksystem.

Zur Datenbank-Engine selber zählen folgende Features:

- Volltextsuche
- Datenbankreplikation

Die Zusatzprodukte, oft unter dem Begriff *Business Intelligence (BI)* zusammengefasst, sind folgende Dienste:

- *Integration Services*: Die Integration Services (IS) sind ein umfassendes Werkzeug, um zum Beispiel Daten von A nach B zu transferieren. Dabei sind komplexe Workflows mit Verzweigungen und unzähligen Möglichkeiten realisierbar.

- *Reporting Services*: Aufgabe dieser Services ist es, Berichte, die auf Daten aus der Datenbank basieren, in verschiedenen Formen zur Verfügung zu stellen. Das kann zum Beispiel eine HTML-Seite oder ein PDF-Dokument sein, das per E-Mail verschickt wird. Ziel ist es, das gesamte Berichtswesen eines Unternehmens abbilden zu können. Daher sind diese Berichte auch nicht statisch. Vielmehr erlauben sie es einem Benutzer, durch die Eingabe von Parametern das Ergebnis zu verändern oder über einen definierten Drill-Down immer detailliertere Daten abzurufen. Ein wichtiger Bestandteil der Reporting Services ist neben der Berichtserstellung die Berichtsverteilung. Reporting Services lassen sich sehr gut in Share Point integrieren.
- *Analysis Services*: Diese dienen der Realisierung von Data-Warehouse-Lösungen. Geschäftsleitung, Controller und Marketingmanager benötigen immer anspruchsvollere Analysen und Trendinformationen. Die Basis dafür liegt zu einem Großteil in den bereits auf Servern gespeicherten Unternehmensdaten. In der Praxis werden zur Lösung dieser Aufgabenstellung OLAP-Systeme (Online Analytical Processing; deutsch: analytische Online-Verarbeitung) benötigt, indem auf einfache Weise Informationszusammenstellungen aus OLTP-Daten erstellt werden, die dann für anspruchsvolle Datenanalysen genutzt werden können. Die Analysis Services bieten diese Funktionalität auf einem sehr hohen Niveau und haben den SQL Server in diesem Bereich zu einem der führenden Produkte gemacht.
- *Service Broker*: Dieser Service zielt auf große verteilte Anwendungen ab. Der Service Broker verwaltet Warteschlangen, die mit SQL-Anweisungen „gefüttert“ werden können. Die Inhalte der Warteschlange werden dann der Reihe (englisch: queue) nach abgearbeitet. Diese Warteschlangen können nicht nur am lokalen Server positioniert sein, sondern auch remote abgearbeitet werden.

Anwendungen, die auf dem Prinzip von Warteschlangen basieren, setzen auf einem anderen Anwendungsverständnis auf, als wir es in der Regel gewohnt sind. Schauen wir uns folgendes Beispiel an: Viele von Ihnen haben sicher schon einmal eine Domänenregistrierung vorgenommen. Wenn Sie eine Domäne registrieren möchten, ist der erste Schritt üblicherweise, dass Sie ermitteln, ob die gewünschte Domäne noch verfügbar ist. In einer Online-Applikation würden Sie eine Schaltfläche anklicken, und die Domäne würde Ihnen gehören. So einfach ist es aber bekanntlich nicht. Sie reichen stattdessen den Antrag bei einer akkreditierten Registrierungsstelle ein. Und hier kommt die Warteschlange ins Spiel. Alle Ihre Eingaben (unter Umständen auch Zusatzinformationen) werden in eine Warteschlange eingereiht. Ihr Antrag steht in der Warteschlange und wird, sobald er an der Reihe ist, bearbeitet. Falls Sie der Erste in der Reihe für diese Domäne gewesen sind, werden Sie die Domäne zugeteilt bekommen.

- *Master Data Services*: Darunter versteht man, wenn Organisationen ihre Stammdaten unternehmensweit zentralisiert vereinheitlichen und für gezielte Analysen bereitstellen.
- *Data Quality Services*: Dies ist ein in dieser Version neues Tool, mit dessen Hilfe die Datenqualität in bestehenden Systemen verbessert werden kann. Lücken in Datenbeständen können damit besser aufgefunden und bereinigt werden. Dies können Fragestellungen sein wie: „Sind alle notwendigen Relationen vorhanden und gesetzt?“

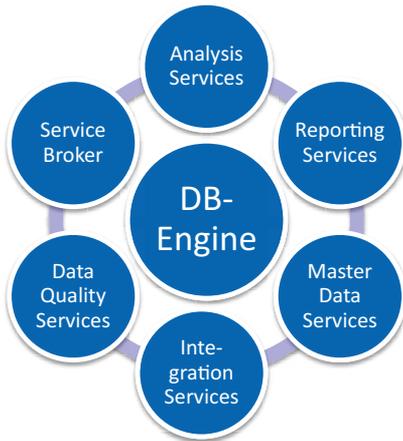


Bild 1.8 SQL Server und seine Zusatzkomponenten

Natürlich stehen diese Features nicht in jeder der verschiedenen SQL Server 2016-Editionen zur Verfügung:

- *Integration Services* stehen ab der Standard Edition zur Verfügung, manche spezielle Formen der Datentransformation erst mit der Enterprise Edition. In allen Editionen ist der SQL Server Import und Export Wizard enthalten.
- *Reporting Services* sind bis zu einem bestimmten Grad bereits ab der Express Edition integriert. Volle Integration findet erst ab der Standard Edition statt.
- *Analysis Services* sind teilweise ab der Standard Edition verfügbar, eine volle Integration ist erst mit der Enterprise Edition gegeben.

■ 1.2 Versionen und Editionen des SQL Servers

Dem Buch liegt die aktuelle Version SQL Server 2017 zugrunde. Diese Version weist gemeinsam mit dem SQL Server 2016 gegenüber deren Vorgängersystem SQL Server 2014 wesentliche Neuerungen auf. Dies betrifft nicht nur die eigentliche relationale Datenbank-Engine, die den Kern des Produkts ausmacht, sondern umfangreiche Erweiterungen der Rahmenprodukte. Diese unter dem Begriff BI (Business Intelligence) zusammengefassten Produkte enthalten beispielsweise die Analysis Services, Integration Services oder Reporting Services.

Erneuerungen gibt es in fast allen Bereichen des SQL Servers. Drei große Schwerpunkte, die ich in meiner täglichen Arbeit nutze, sind:

- *Speicheroptimierte Tabellen*: Ein wahrer Performanceboost sind die mit dem SQL Server 2014 eingeführten speicheroptimierten Tabellen. Hinter diesem Begriff verbirgt sich die Möglichkeit, ganze Tabellen vollständig im RAM zu halten. Da der Zugriff auf Festplatten in der Regel das ist, was eine Datenbank am meisten bremst, bietet diese Möglichkeit

ungeahnte Performancesteigerungen gegenüber herkömmlichen Tabellen. Hat es in der ersten Implementierung noch einige Einschränkungen in der Verwendung gegenüber herkömmlichen Tabellen gegeben – was in der ersten Implementierung eines komplexen und bahnbrechenden Features nichts Ungewöhnliches ist –, sind die meisten dieser Limitationen nun ausgeräumt. Damit ist dies nun ein extrem gutes und praxistaugliches Feature. Und ein weiteres Plus aus meiner Sicht: Es ist nun nicht mehr auf die Enterprise Edition beschränkt.

- *Always Encrypted*: Sensible Daten werden durch Zugriffsrechte auf dem Server vor unbefugtem Zugriff geschützt. Schon bisher hat es die Möglichkeit gegeben, Zugriffsrechte dermaßen einzuschränken, dass Daten vor unbefugten Zugriffen bestmöglich geschützt worden sind. Daten im Backup zu verschlüsseln, ergänzte diese Möglichkeiten, da die Zugriffsrechte, sobald die Datenbank auf einem eigenen Server eingespielt werden kann, selbst definiert werden können. Mit den neuen Möglichkeiten sind sie auch auf dem Server in der Datenbank verschlüsselbar, sodass selbst jemand, der sich der gesamten Datenbank bemächtigt, diese Daten nicht mehr einsehen kann.
- *Temporale Tabellen*: Änderungsprotokollierung ist in der Praxis sehr oft ein Thema. Trotz mehrerer Ansätze in der Vergangenheit, die in der Praxis kein voll befriedigendes Ergebnis geliefert haben, dies einfach vom System her zur Verfügung zu stellen, werden saubere und zufriedenstellende Lösungen mit Triggern ausprogrammiert. Temporale Tabellen liefern aus meiner Sicht erstmals eine sehr brauchbare Lösung, ohne eigene Programmierung eine Lösung für diese Aufgabenstellung zu erhalten. Mit dem SQL Server 2016 eingeführt, unterstützen sie mit dem SQL Server 2017 auch Änderungs- und Löschesweitergaben bei Beziehungen (FOREIGN KEY mit DELETE CASCADE und UPDATE CASCADE).
- *Strict Security für Common Language Runtime (CLR)*: Neue Erweiterungen für die Sicherheit von CLR-Code erhöhen mit dem SQL Server 2017 zwar den Aufwand, um CLR-Code auf dem SQL Server ausführen zu können, bieten aber deutlich höhere Sicherheit. Unter CLR-Code versteht man mit C# oder VB.NET programmierte Prozeduren, die mittels der Common Language Runtime (CLR) direkt auf dem SQL Server ausgeführt werden können.

Ein paar weitere Erweiterungen, die teilweise auch in diesem Buch behandelte Themen betreffen, habe ich exemplarisch in der nachfolgenden Tabelle angeführt.

Tabelle 1.1 Einige Neuerungen in SQL Server 2016 und 2017

Thema	Beschreibung
Windows Azure	Daten können, anstelle lokal auf dem Server gehalten zu werden, in Windows Azure-BLOBs abgelegt werden. Das Hosten von SQL Server-Datenbanken auf einem virtuellen Computer in Windows Azure wird über eigene Bereitstellungsassistenten unterstützt. Die Sicherung einer SQL Server-Datenbank kann über eine URL in Windows Azure-BLOBs erfolgen.
Erstellen von Ausführungsplänen	Durch die überarbeitete Logik der Kardinalitätsschätzung werden die Qualität und damit die Effizienz von Ausführungsplänen verbessert. Das wiederum steigert die Abfrageleistung. (Ausführungspläne legen fest, wie der SQL Server intern eine von uns getätigte Abfrage abarbeitet.)
Live-Abfragestatistik	Abfragestatistiken zur Laufzeit bieten Administratoren einen tieferen Einblick in die Verarbeitung von Anweisungen.
Transact-SQL-Erweiterungen	Es gibt zahlreiche Erweiterungen in Form von neuen Funktionen und Anweisungen in der Sprache zur Bearbeitung von Daten. Beispielsweise sind dies die Funktionen DATEDIFF_BIG(), STRING_SPLIT() und STRING_ESCAPE() mit dem SQL Server 2016 und die Funktionen STRING_AGG(), TRANSLATE(), CONCAT_WS() und TRIM() mit dem SQL Server 2017.
Sicherheits-erweiterungen	Berechtigungen können nun auf Datensatzebene vergeben werden. Für diese Funktionalität ist bisher ein Workaround über gefilterte Sichten notwendig gewesen.
Temporäre Datenbank <i>tempdb</i>	Das Aufteilen der temporären Datenbank <i>tempdb</i> auf mehrere Dateien, die auch auf unterschiedlichen Datenträgern verwaltet werden können, ermöglicht eine bessere interne Nutzung und damit eine bessere Performance.
Native JSON-Unterstützung	Für Entwickler interessant ist die native Unterstützung von JSON als Ausgabeformat mit der Klausel FOR JSON, die mit der bisherigen Klausel FOR XML vergleichbar ist.
UTF-8 Unterstützung	Ich bin geneigt zu sagen „endlich“! Die Anweisung BULK INSERT, mit der Textdateien (CSV) einfach und schnell importiert werden können, unterstützt ab dem SQL Server 2016 nun auch UTF-8. Das ist mir in der Vergangenheit sehr oft abgegangen.
GRAPH-Unterstützung	Die Unterstützung von Graph-Datenbanken des SQL Server 2017 ist ein weiterer Schritt zur Integration von NoSQL-Elementen.

Editionen des SQL Server 2017

Microsoft liefert den SQL Server 2017 in einer Reihe unterschiedlicher Editionen aus. Ziel dieser Produktdifferenzierung ist es, dem Kunden ein Angebot zu unterbreiten, das es ermöglicht, den jeweiligen Anforderungen in Hinblick auf Leistungsfähigkeit, Laufzeit und Preise gerecht zu werden. Darüber hinaus werden zahlreiche Zusatzkomponenten angeboten. Welche dieser Komponenten im Einzelfall für eine Installation ausgewählt werden, hängt von den konkreten Anforderungen ab. Die früher verfügbare Business Intelligence Edition ist seit dem SQL Server 2016 nicht mehr verfügbar.

5

Transact-SQL – die Sprache zur Serverprogrammierung

In diesem Kapitel lernen Sie die grundlegenden Konzepte der Datenbankprogrammiersprache Transact-SQL. Sie erfahren, aus welchen Bestandteilen diese Sprache besteht, wie sie eingesetzt wird und welcher Syntax sie sich bedient. Am Ende dieses Kapitels sollten Sie in der Lage sein, die Sprache zu verstehen und anzuwenden. Das Erstellen von Datenbankobjekten, die mittels dieser Sprache programmiert werden, ist Thema des darauffolgenden Kapitels.

Über die SQL Server CLR (Common Language Runtime) kann serverseitige Programmierung auch mit .NET-Programmiersprachen erfolgen. Dies ist eine Ergänzung zu Transact-SQL, um gemeinsam das gesamte Spektrum der Datenbankprogrammierung abzudecken. Sie können auch als .NET-Programmierer nicht auf Transact-SQL verzichten. Spätestens dann, wenn Sie auf Daten zugreifen, werden Sie diese Anweisungen auch innerhalb des .NET-Codes benötigen. Die Einsatzbereiche dieser beiden Sprachen lassen sich wie folgt abgrenzen:

- Transact-SQL sollten Sie immer dann einsetzen, wenn Datenzugriffe im Vordergrund stehen. Dabei spielt es keine Rolle, ob Sie lesend oder schreibend auf Daten zugreifen. Effiziente Formen des Datenzugriffs sind die Stärke von Transact-SQL.
- .NET-Programmierung ist dann zu bevorzugen, wenn es aufwendige Algorithmen umzusetzen gilt. Zusätzlich wird .NET-Programmierung für alles eingesetzt, was nicht direkt datenbankspezifisch ist; zum Beispiel der Zugriff auf das Filesystem, auf einen FTP-Server oder das Bearbeiten einer Grafikdatei, bevor sie in die Datenbank eingefügt wird.



HINWEIS: In diesem Kapitel beschäftigen wir uns ausschließlich mit Transact-SQL. Der .NET-Programmierung mit dem SQL Server widmen wir uns später in einem separaten Kapitel.

Die standardisierte Abfragesprache SQL (ANSI SQL) ist keine prozedurale Sprache. Zur Programmierung reichen die vorhandenen Funktionalitäten nicht aus. Weiterführende Funktionen werden von der prozeduralen Spracherweiterung von SQL geboten: Transact-SQL.

Transact-SQL ist eine prozedurale Spracherweiterung zu SQL, die bestimmte Konstrukte aufweist, die wir von Programmiersprachen der dritten Generation (3GL-Sprachen) her kennen. Genannt seien hier beispielsweise Auswahl- und Wiederholungsstrukturen. Die Erwei-

terung ist rein herstellerbezogen und nur in den Produkten MS SQL Server und Sybase Adaptive Server Enterprise enthalten. Das liegt daran, dass diese beiden Produkte vor vielen Jahren gemeinsame Wurzeln gehabt haben. Gleichartige und vergleichbare Sprachen haben auch andere Hersteller in ihren Datenbankprodukten implementiert. Beispielsweise heißt die prozedurale Spracherweiterung bei Oracle PL/SQL, wobei PL für Procedural Language steht. PL/SQL und Transact-SQL haben jedoch – außer der Grundkonzeption und Funktionalität – hinsichtlich ihrer Sprachsyntax nicht allzu viele Gemeinsamkeiten.

Transact-SQL – auch kurz als T-SQL bezeichnet – wird bei der Entwicklung von Datenbanken unter anderem für die Erstellung *gespeicherter Prozeduren* (Stored Procedures), *benutzerdefinierter Funktionen* (Userdefined Functions) und *Schalter* (Trigger) verwendet. Zunächst zur begrifflichen Abgrenzung:

1. *Stored Procedures* sind Programme, die direkt auf dem Server gespeichert und ausgeführt werden. Der Aufruf erfolgt häufig aber von Client-Anwendungen aus. Um die Wirkungsweise der Prozedur zu steuern, werden einer Stored Procedure beim Aufruf Parameterwerte übergeben. Dies erfolgt wie bei Prozeduren in anderen Programmiersprachen auch.
2. *Userdefined Functions* liefern wie Systemfunktionen einen Wert oder eine Tabelle zurück und können im Gegensatz zu gespeicherten Prozeduren auch in SQL-Anweisungen verwendet werden.
3. *Trigger* sind mit den Ereignisprozeduren in anderen Programmiersprachen vergleichbar. Sie können nicht wie Prozeduren und Funktionen explizit aufgerufen werden. Sie werden vielmehr durch Ereignisse ausgelöst, die in Tabellen auftreten. Diese sind das Einfügen (INSERT), Ändern (UPDATE) und Löschen (DELETE) von Datensätzen.

Auf das Erstellen von gespeicherten Prozeduren, benutzerdefinierten Funktionen und Triggern gehe ich im nächsten Kapitel ein. In diesem Kapitel werden wir uns der Sprache T-SQL widmen.

Für die Eingabe und Erstellung der Beispiele in diesem Kapitel verwenden wir das SQL Server Management Studio. Öffnen Sie dieses bitte, melden Sie sich am Datenbankmodul Ihres Servers an und wählen Sie die Datenbank *wawi* aus.

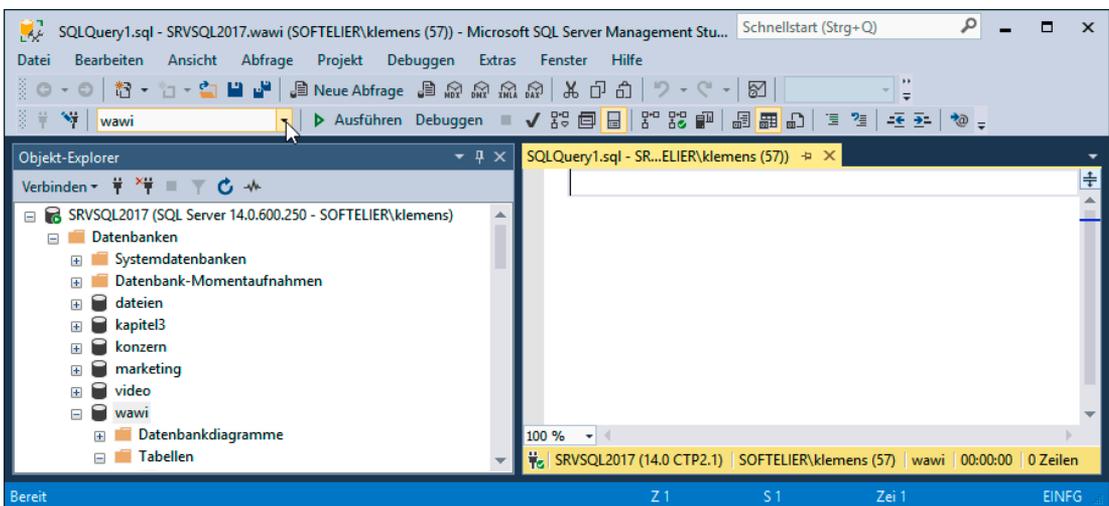


Bild 5.1 Beispieldatenbank für die Arbeit in diesem Kapitel auswählen

■ 5.1 Bestandteile und Funktionalität von Transact-SQL

In diesem ersten Abschnitt gebe ich Ihnen eine Übersicht über die wichtigsten Elemente von T-SQL.



HINWEIS: Damit Sie die nachfolgenden Beispiele verstehen, möchte ich darauf hinweisen, dass in Transact-SQL so wie auch in SQL innerhalb einer Anweisung ein Zeilenumbruch stattfinden darf. Es kann also vorkommen, dass eine Programmzeile nicht unbedingt eine Codezeile sein muss. Solche Zeilenumbrüche werden vor allem dann verwendet, wenn durch sie die Lesbarkeit besonders langer Anweisungen erhöht werden kann. Sehr häufig betrifft dies SQL-Anweisungen, bei denen jede Klausel in eine eigene Zeile geschrieben wird.

5.1.1 Variablen und Datentypen

Wie in jeder anderen Programmiersprache stehen in Transact-SQL auch Variablen zur Verfügung. Es handelt sich dabei um benutzerdefinierte Objekte, die einen Datentyp haben und in denen Werte während der Programmausführung zwischengespeichert und abgerufen werden können. Diese können zum einen Variablen sein, die im Programmcode deklariert werden und bei der Programmausführung Werte zugewiesen bekommen. Es können zum anderen auch Variablen sein, die bereits beim Aufruf der Prozedur mit Übergabewerten gefüllt werden.

Einer Variablen können nicht beliebige Inhalte zugewiesen werden. Diese besitzen – genauso wie Datenfelder einer Tabelle – bestimmte Datentypen. Die für Variablen verwendbaren Datentypen sind dieselben, die innerhalb der Datenbank für Felddatentypen zur Verfügung stehen.

Eine Übersicht entnehmen Sie der nachfolgenden Tabelle.

Tabelle 5.1 Datentypen für Variablen

Kategorie	Datentyp
Character	CHAR(Länge)
	VARCHAR(Länge)
	NCHAR (Länge)
	NVARCHAR(Länge)
	VARCHAR(MAX)
	NVARCHAR(MAX)

(Fortsetzung nächste Seite)

Tabelle 5.1 Datentypen für Variablen (*Fortsetzung*)

Kategorie	Datentyp
Datum/Uhrzeit	DATETIME SMALLDATETIME DATE DATETIME2(Länge) DATETIMEOFFSET(Länge) TIME
Zahl	DECIMAL(Genauigkeit, Dezimalstellen) FLOAT(Länge) REAL BIGINT INT SMALLINT TINYINT
Währung	MONEY SMALLMONEY
Boolean	BIT
Binär	BINARY(Länge) VARBINARY(Länge) VARBINARY(MAX)
XML	XML
Variante	SQL_VARIANT
Hierarchie	HIERARCHYID
Räumlich	GEOGRAPHY GEOMETRY

Da Sie vielleicht so manche SQL Server-Version „übersprungen“ haben, möchte ich noch kurz auf den Datentyp *VARCHAR(MAX)* eingehen. Dieser Datentyp vereint die Vorteile der Datentypen *VARCHAR()* und *TEXT*. In einem Feld und in einer Variablen vom Datentyp *TEXT* können mehr als 8000 Zeichen gespeichert werden; allerdings können diese nicht mit Standard-SQL-Anweisungen und Zeichenfolgefunktionen verarbeitet werden. *TEXT* und *IMAGE* – ersetzt durch *VARBINARY(MAX)* – existieren nur aus Gründen der Abwärtskompatibilität und sind bereits auf der Liste der abgekündigten Features.

Folgendes können Sie mit *VARCHAR(MAX)* tun, was mit *TEXT* nicht möglich ist:

- Verwenden als Datentyp für Variablen
- Verwenden von String-Funktionen wie beispielsweise *CHARINDEX()* oder *REPLACE()* zur Bearbeitung
- Inhalte mit anderen Feldern oder Variablen eines *CHARACTER*-Datentyps verketteten

Auch *IMAGE* kann nicht für Variablen verwendet werden, der Nachfolger *VARBINARY(MAX)* aber sehr wohl.

Beachten Sie bitte, dass SQL Server zwischen zwei Arten von Variablen unterscheidet:

1. *Benutzerdefinierte Variablen* werden innerhalb eines Transact-SQL-Programms oder einer Benutzersitzung vom Benutzer erzeugt und gelten ausschließlich innerhalb des Pro-

gramms oder der Sitzung, in der sie deklariert wurden. Benutzerdefinierte Variablen werden innerhalb einer Prozedur mit der Anweisung DECLARE erzeugt. Der Name von benutzerdefinierten Variablen beginnt stets mit einem @.

2. *Globale Variablen* sind vom System vordefinierte Variablen, deren Inhalte durch das System zugewiesen werden. Die Inhalte dieser Variablen geben dem Benutzer wertvolle Informationen über das System oder über aktuelle Zustände im Programmcode. Die Namen von globalen Variablen beginnen immer mit @@. Globale Variablen können nur gelesen werden; ihnen kann explizit kein Wert zugewiesen werden. Damit ähneln sie Systemfunktionen in der Verwendung.



HINWEIS: Das System, dass Variablen in Transact-SQL immer mit einem @ beginnen, erleichtert die Arbeit und vor allem die Lesbarkeit von Programmcode enorm. Variablen sind sofort als solche zu erkennen, auch wenn sie denselben Namen haben wie das Feld einer Tabelle. Verwechslungen mit Feldnamen sind daher ausgeschlossen. Nur zum Vergleich: In der Oracle-Programmiersprache PL/SQL werden Variablen nicht extra derart als solche gekennzeichnet. Hier muss sich der Entwickler an Prioritätsregeln halten, um zu bestimmen, ob bei Namensgleichheit die Variable oder der Feldname gemeint ist. In dieser Hinsicht gefällt mir die SQL Server-Implementierung wesentlich besser.

Lokale Variablen deklarieren

Lokale Variablen werden mit der Anweisung DECLARE unter Angabe ihres Datentyps definiert. Dabei kann optional das Schlüsselwort AS verwendet werden.

```
DECLARE @var1 AS int
DECLARE @var2 AS smalldatetime
DECLARE @var3 AS varchar(25)
```

Mit einer DECLARE-Anweisung können Sie auch mehrere Variablen in einer Zeile deklarieren. Dabei müssen alle Variablen mit Komma voneinander getrennt geschrieben werden.

```
DECLARE @var1 int, @var2 smalldatetime, @var3 varchar(25)
DECLARE @var4 int
```



ACHTUNG! Auch wenn Sie mehrere Variablen desselben Datentyps deklarieren, muss bei jeder Variablen der Datentyp separat angegeben werden. Es ist nicht möglich, eine Auswahl für mehrere Variablen gemeinsam zu definieren.

So ist zum Beispiel nachfolgende Deklaration, die drei Variablen vom Typ Integer deklarieren soll, ungültig.

```
DECLARE @var1, @var2, @var3 int
```

Stattdessen muss der Datentyp bei jeder Variablen explizit angegeben werden.

```
DECLARE @var1 int, @var2 int, @var3 int
```

Die Wertzuweisung an eine Variable kann auf zwei Arten erfolgen:

- SET-Anweisung
- SELECT-Anweisung

Die direkte Zuweisung eines Variablenwertes erfolgt mit der Anweisung SET. Die Syntax hierzu lautet:

```
SET @variable = wert
```

Als Wert kann der Variablen ein skalarer Wert, ein Berechnungsausdruck oder das Ergebnis einer Unterabfrage zugewiesen werden. Dabei ist zu berücksichtigen, dass die Unterabfrage in runde Klammern gesetzt werden muss.

```
SET @variable = (SELECT wert FROM ...)
```

Wird ein Wert über eine Unterabfrage zugewiesen, muss diese so ausgelegt sein, dass sie genau eine Spalte und eine Zeile zurückgibt. Liefert die Abfrage mehrere Zeilen – weil beispielsweise die WHERE-Klausel nicht korrekt ist –, führt dies zu einem Fehler.

Sie deklarieren beispielsweise eine Variable und weisen ihr den Namen eines Mitarbeiters zu. Sie vergessen dabei aber die WHERE-Klausel, die sicherstellen sollte, dass die Unterabfrage nur eine Zeile zurückliefert. Also zum Beispiel so:

```
DECLARE @nachname varchar(50)
SET @nachname = (SELECT nachname FROM dbo.personal)
```

Das System meldet Ihnen einen Fehler:

```
Meldung 512, Ebene 16, Status 1, Zeile 2
Die Unterabfrage hat mehr als einen Wert zurückgegeben. Das ist nicht zulässig, wenn
die Unterabfrage auf =, !=, <, <=, > oder >= folgt oder als Ausdruck verwendet wird.
```

Um mehrere Werte aus einer Tabelle abzufragen, müssen Sie daher mehrere SET-Anweisungen verwenden.

Sie möchten beispielsweise den Nachnamen, den Vornamen und das Geburtsdatum des Mitarbeiters mit der Personalnummer 452 in Variablen einlesen:

```
DECLARE @nachname varchar(50), @vorname varchar(50)
DECLARE @gebdatum date

SET @nachname = ( SELECT nachname
                  FROM dbo.personal
                  WHERE persnr = 452 )
SET @vorname = ( SELECT vorname
                 FROM dbo.personal
                 WHERE persnr = 452 )
SET @gebdatum = ( SELECT gebdatum
                  FROM dbo.personal
                  WHERE persnr = 452 )

SELECT @nachname AS NN, @vorname AS VN, @gebdatum AS Geburtsdatum;
```

Ergebnis:

NN	VN	Geburtsdatum
Kossegg	Anita	1969-06-20

(1 Zeile(n) betroffen)

Die abschließende SELECT-Anweisung dient zur Anzeige der Variableninhalte.

Da Sie mit der SET-Anweisung immer nur einen Wert zuweisen können, müssen Sie mehrere Anweisungen und damit mehrere Abfragen hintereinander verwenden, um drei Werte aus derselben Zeile einer Tabelle auszulesen und in Variablen abzulegen.

In einer solchen Situation ist es effizienter und sinnvoller, die SELECT-Anweisung zur Zuweisung der Variableninhalte zu verwenden. Da mit einer SELECT-Anweisung auch mehrere Variablen mit einer einzigen Anweisung befüllt werden können, benötigen Sie für das vorangegangene Beispiel anstelle von drei separaten Datenzugriffen lediglich einen einzigen.

Die Syntax für die Wertzuweisung über die SELECT-Anweisung lautet:

```
SELECT @var1 = wert1, @var2 = wert2, @var3 = wert3, ...
[FROM ...]
```

Jeder Variablen wird ein Wert zugewiesen. Die einzelnen Zuweisungen werden voneinander mit Komma getrennt. Sofern die Werte aus einer Abfrage stammen, kann diese direkt in die Zuweisung integriert werden. Ergänzen Sie dazu die SELECT-Anweisung mit einer FROM-Klausel und optional mit weiteren Klauseln, die Sie von SELECT-Anweisungen her kennen.

Das obige Beispiel (Name, Vorname und das Geburtsdatum für den Mitarbeiter mit der Personalnummer 452 sollen in einer Variablen gespeichert werden) ist mithilfe der SELECT-Anweisung folgendermaßen zu realisieren:

```
DECLARE @nachname varchar(50), @vorname varchar(50)
DECLARE @gebdatum date

SELECT @nachname = nachname,
       @vorname = vorname,
       @gebdatum = gebdatum
FROM dbo.personal
WHERE persnr = 452;

SELECT @nachname AS NN, @vorname AS VN, @gebdatum AS Geburtsdatum;
```

Ergebnis:

NN	VN	Geburtsdatum
Kossegg	Anita	1969-06-20

(1 Zeile(n) betroffen)

Auch hier ist darauf zu achten, dass die Anweisung nur eine Zeile zurückliefert. Im Unterschied zur ersten Variante führt es aber zu keinem Fehler, falls mehrere Zeilen geliefert werden. Nach der Anweisung sind jene Werte in den Variablen anzufinden, welche die letzte zurückgegebene Zeile geliefert hat. Da dies oft zu unerwarteten Ergebnissen führen kann,

7

SQL Server CLR-Integration

Mittlerweile schon seit der Version 2005 sind Sie beim Programmieren des SQL Servers nicht mehr nur auf Transact-SQL beschränkt. Dabei arbeiten der SQL Server und das Visual Studio eng zusammen. Das ist auch notwendig, denn der SQL Server enthält eine *Common Language Runtime* (CLR), das heißt, er ist in der Lage, .NET-Code laufen zu lassen. Für die Code-Entwicklung benötigen Sie das Visual Studio. Erst der fertige Code wird in einer Datenbank auf dem SQL Server integriert.

Die *SQL Server Data Tools* (SSDT) sind einerseits in Visual Studio 2015 und 2017 integriert, sind aber andererseits auch in beiden Versionen frei separat verfügbar. Die Integration erstreckt sich auch auf die freie Community Edition des Visual Studio. Somit ist gewährleistet, dass man ohne separat lizenziertes Visual Studio Professional den vollen Umfang der CLR-Programmierung für den SQL Server nutzen kann.

Die Data Tools sind ein umfangreiches Toolset, das alle Bereiche der Datenbankentwicklung mit dem SQL Server abdeckt. Dies ist nicht nur für Programmierer von Vorteil, die lieber in ihrer gewohnten Umgebung bleiben möchten und nicht so gerne mit den SQL Server-Tools arbeiten. Wir konzentrieren uns in diesem Kapitel auf die Programmierung für die SQL Server CLR und werden uns im nächsten Kapitel ausführlicher mit den Data Tools auseinandersetzen.

Sofern Sie noch keine Form der Data Tools auf Ihrem Rechner verfügbar haben, laden Sie diese unter folgender Adresse herunter: <https://docs.microsoft.com/de-de/sql/ssdt/download-sql-server-data-tools-ssdt>

Je nachdem, welche Version Sie bevorzugen, laden Sie sich die SSDT für Visual Studio 2015 oder 2017 herunter oder eine der unterstützten Editionen für Visual Studio in denselben Versionen. Die SQL Server Data Tools integrieren sich in eine bereits installierte Version oder verwenden die abgespeckte Visual Studio Shell. Ich verwende hier das freie Visual Studio 2017 Community Edition. Für Sie sollte es einerlei sein, ob Sie Visual Studio 2015 oder 2017 einsetzen.



ACHTUNG! Frühere Versionen der SQL Server Data Tools werden nicht mehr aktualisiert und sind bereits nicht mehr auf dem letzten Stand. Daher können Sie diese für den SQL Server 2017 nicht mehr verwenden.

Ziel der CLR-Programmierung ist es, Transact-SQL in denjenigen Bereichen zu ergänzen, wo es naturgemäß Schwächen gibt. Dies sind vor allen Aufgabenstellungen, die

- einen sehr komplexen Algorithmus verlangen
- oder einen Bezug außerhalb der Datenbank – wie zum Beispiel Zugriff auf das Dateisystem – aufweisen.

Überall dort, wo Datenzugriffe im Vordergrund stehen, sollte weiterhin T-SQL zum Einsatz kommen. Aufgabenstellungen, für die in älteren SQL Server-Versionen erweiterte Systemprozeduren (xp_...) zum Einsatz kamen, werden nun über eine Common Language gelöst. Erweiterte Systemprozeduren werden lediglich aus Gründen der Abwärtskompatibilität noch unterstützt.



HINWEIS: Für die Arbeit mit diesem Kapitel ist es von Vorteil, wenn Sie bereits mit der .NET-Programmierung und dem Visual Studio vertraut sind. Insbesondere benötigen Sie Kenntnisse in ADO.NET. Da eine eingehende Behandlung dieser Themen über den Rahmen dieses Buches hinausginge, verweise ich auf weiterführende Literatur zu diesen Themen.

■ 7.1 Mit im Boot: .NET Framework

Der SQL Server ist durch die Common Language Runtime (CLR) in der Lage, .NET-Code auszuführen. Das Visual Studio dient als Entwicklungswerkzeug für die vom SQL Server ausführbaren Objekte.

Diese sind:

- .NET User-Defined Functions (UDF)
- .NET Stored Procedures
- .NET Trigger
- User-Defined Aggregates (UDA)
- User-Defined Datatypes (UDT)

Benutzerdefinierte Funktionen, gespeicherte Prozeduren und Trigger gleichen in ihrer Funktionalität und ihrem Einsatzbereich ihren Transact-SQL-Pendants.



HINWEIS: Microsoft bedient sich bei der Weiterentwicklung des SQL Servers dieser Funktionalität. So sind die räumlichen Datentypen *geography* und *geometry* sowie der Datentyp *hierarchy_id* als .NET-Datentypen integriert worden.

Aber damit ist noch nicht alles abgedeckt – das .NET Framework spielt auch bei der Verwaltung des SQL Servers mit. Die COM-basierten *Distributed Management Objects (SQL-DMO)*,

die früher verwendet worden sind, sind mittlerweile durch die .NET-basierten *SQL Server Management Objects (SMO)* abgelöst worden.

Mit dem .NET Framework und der Datenbank-Engine prallen aber auch zwei Welten aufeinander, zwischen denen Brücken geschlagen werden müssen. Daher gibt es einen eigenen Satz an SQL-Datentypen, um SQL Server-Datentypen mit den .NET-Datentypen zu verbinden. Eine Übersicht finden Sie in der nachfolgenden Tabelle.

Innerhalb von .NET-Code verwenden Sie die SQL-Typen an den Schnittstellen von und zur Datenbank. Innerhalb des Codes verwenden Sie wie gewohnt die .NET-Datentypen. Sie benutzen übliche Konvertierungen, um Inhalte von Variablen mit .NET-Typen in solche mit SQL-Typen und umgekehrt zu konvertieren.

Tabelle 7.1 Datentypenzuordnung

SQL Server-Datentyp	SqlType	.NET-Datentyp
char varchar nchar nvarchar text ntext	SqlString	String
bigint	SqlInt64	Int64
int	SqlInt32	Int32
smallint	SqlInt16	Int16
tinyint	SqlByte	Byte
numeric decimal	SqlDecimal	Decimal
money smallmoney	SqlMoney	Decimal
real	SqlSingle	Single
float	SqlDouble	Double
datetime smalldatetime	SqlDateTime	Datetime
bit	SqlBoolean	Boolean
binary varbinary image timestamp	SqlBinary SqlBytes	Byte()
uniqueidentifier	SqlGuid	Guid

Sie sollten, bevor Sie mit diesem Kapitel arbeiten, die Kapitel 5 und 6 gelesen haben. In diesen lernen Sie nicht nur die Sprache Transact-SQL, sondern auch die Konzepte hinter dem Einsatz von gespeicherten Prozeduren, Triggern und benutzerdefinierten Funktionen kennen. Diese werden in diesem Kapitel benötigt und nicht nochmals erarbeitet.

7.1.1 Integration mit dem Visual Studio

Die SSDT werden mit dem Setup von SQL Server noch nicht mit installiert. Wie zuvor und in Kapitel 2 beschrieben, müssen Sie die SQL Server Data Tools oder eine entsprechende Visual Studio-Version separat installieren.



HINWEIS: In diesem Kapitel werden wir die Beispiele mit Visual Basic .NET umsetzen. Sie finden bei den Beispieldateien zum Buch allerdings alle Beispiele auch mit C# umgesetzt. Beide Varianten im Text zu behandeln, würde den zur Verfügung stehenden Rahmen sprengen. Daher habe ich mich für das für Einsteiger einfachere Visual Basic entschieden.

Sie finden bei den Buchbeispielen Projektdateien in VB.NET und C# für die Visual Studio-Versionen 2015 und 2017.

Erstellen wir zu Beginn mit den SSDT ein neues Projekt mit der Vorlage *SQL Server-Datenbankprojekt* aus der Gruppe *SQL Server*.

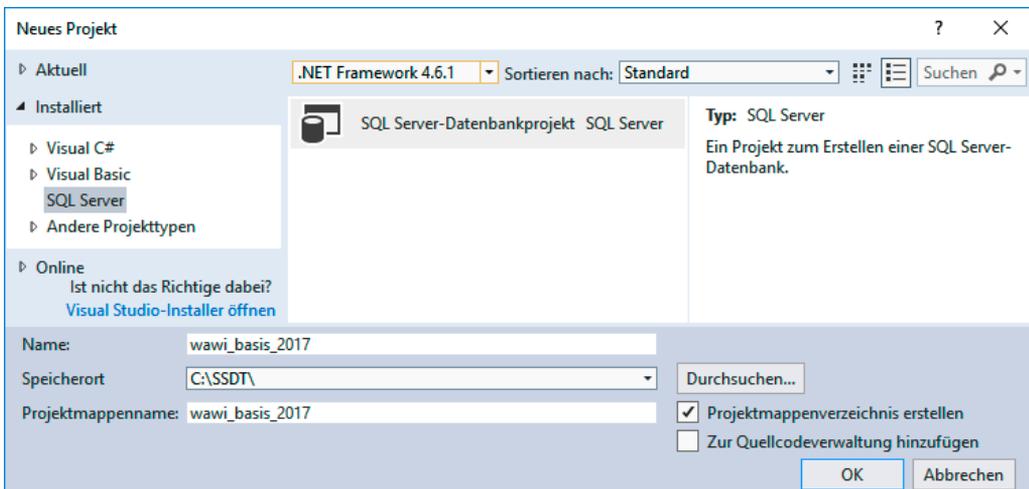


Bild 7.1 Neues SQL Server-Datenbankprojekt in Visual Studio 2017



ACHTUNG! Ändern Sie nach dem Erstellen das Framework von 4.6.1 zum Beispiel auf 4.0, damit das Projekt später auch auf den SQL Server 2012 übertragen werden kann. Verwenden Sie die Version 3.5, wenn Sie auch den SQL Server 2008 R2 unterstützen möchten. In Visual Studio 2017 können Sie dies auch schon direkt im Dialog beim Erstellen erledigen.

Der Name, den Sie diesem Projekt geben, wird später in Ihrer Datenbank als Assembly-Name verwendet, wenn Sie die im Studio erstellten Objekte von Visual Studio automatisch bereitstellen oder veröffentlichen lassen. Das neue Projekt wird wie gewohnt im Projekt-

mappen-Explorer mit dem Projektnamen angezeigt. Dort finden Sie den Eintrag *Projektmappe „wawi_basis_2017“*, den Ordner *wawi_basis_2017* mit den Unterordnern *Properties* und *Verweise*.

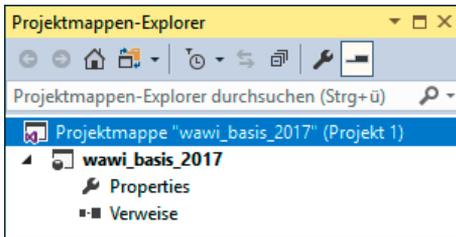


Bild 7.2 Projektmappen-Explorer

In Visual Studio 2015/2017 ist standardmäßig eine Verbindung zu einer LocalDB im SQL Server-Objekt-Explorer eingerichtet. Ist dieser bei Ihnen noch nicht sichtbar, können Sie ihn über das Menü **ANSICHT** einblenden. Wenn Sie möchten, richten wir uns eine neue Verbindung zu unserem Server ein, da wir mit unserer Datenbank *wawi* arbeiten möchten. Wir benötigen diese Verbindung zwar nicht unbedingt, um für die CLR zu programmieren, aber wir haben dadurch die Namen von Tabellen und Spalten im Blickfeld, was sicher kein Nachteil ist. Außerdem können wir erstellte Objekte sofort in der Datenbank sehen. Wählen wir dazu im Menü **EXTRAS** den Befehl **SQL SERVER HINZUFÜGEN...** oder klicken auf das entsprechende Symbol im *SQL Server-Objekt-Explorer*. Im anschließenden Anmeldedialog, den wir vom SQL Server Management Studio kennen, melden wir uns an unserem Server an.

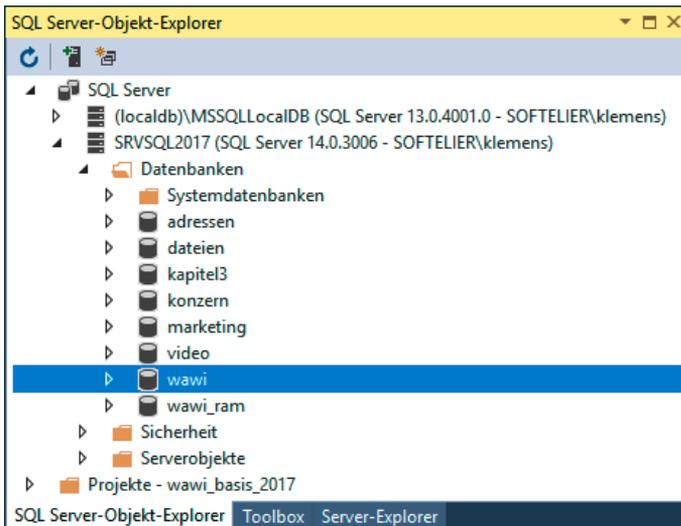


Bild 7.3 SQL Server-Objekt-Explorer



ACHTUNG! Je nachdem, welche Installationsmedien Sie für Ihr Visual Studio verwendet haben, müssen Sie noch ein Update installieren, bevor Sie mit dem SQL Server 2017 arbeiten können. Sie erhalten sonst eine Fehlermeldung beim Versuch, sich mit dem aktuellen SQL Server zu verbinden.

Verwenden Sie den Menübefehl **EXTRAS/EXTENSIONS UND UPDATES...**, um verfügbare Updates anzuzeigen.

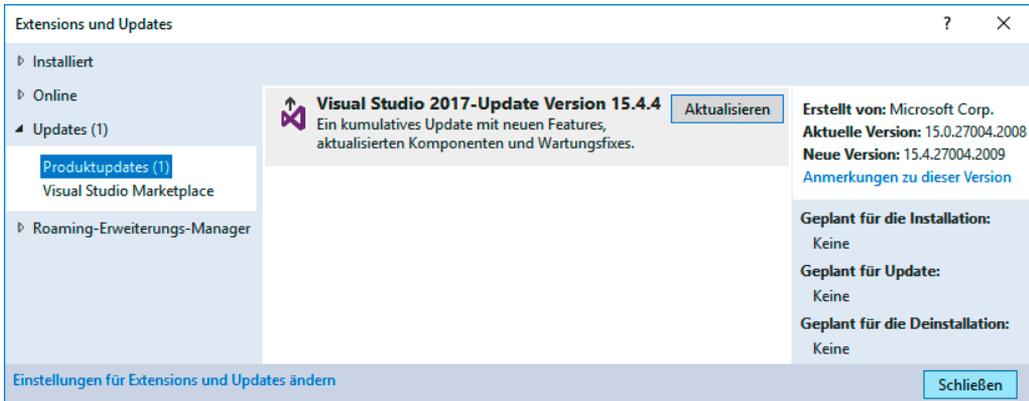


Bild 7.4 Aktualisierungen installieren

Bei der Installation von Visual Studio 2017 ist bereits eine LocalDB mit der Version 13 (SQL Server 2016) installiert worden. Haben Sie auf Ihrem Rechner zuvor auch andere Versionen des Visual Studio installiert, können weitere Versionen der LocalDB vorhanden sein. Wie auch bei den Vorversionen, wird mit einem zukünftigen Update auch die LocalDB in der aktuellsten Version 14 ergänzt werden. Jedenfalls werden alle verfügbaren Instanzen im SQL Server-Objekt-Explorer automatisch angezeigt. In Bild 7.3 erkennen Sie die als Erstes erwähnte Version im SQL Server-Objekt-Explorer über der gerade ergänzten Verbindung.

Projekteinstellungen können unter den Eigenschaften des Projektordners vorgenommen werden. Klicken Sie dazu im Projektmappen-Explorer den Ordner *Properties* doppelt an. Die Eigenschaften sind, wie in Visual Studio üblich, in verschiedene Kategorien unterteilt. Die für uns im Moment wichtigen Einstellungen finden wir unter *SQLCLR*. Hier werden der Name der Assembly und die Berechtigungsstufe – über beides werden wir später noch sprechen – eingestellt. Auch das Zielframework kann hier konfiguriert werden. Damit kann die diesbezügliche Auswahl, die Sie im Visual Studio 2017 beim Erstellen des Projekts getroffen haben, noch einmal verändert werden. Hier stellen Sie auch ein, ob Sie Visual Basic oder C# als Sprache für dieses Projekt einsetzen möchten.

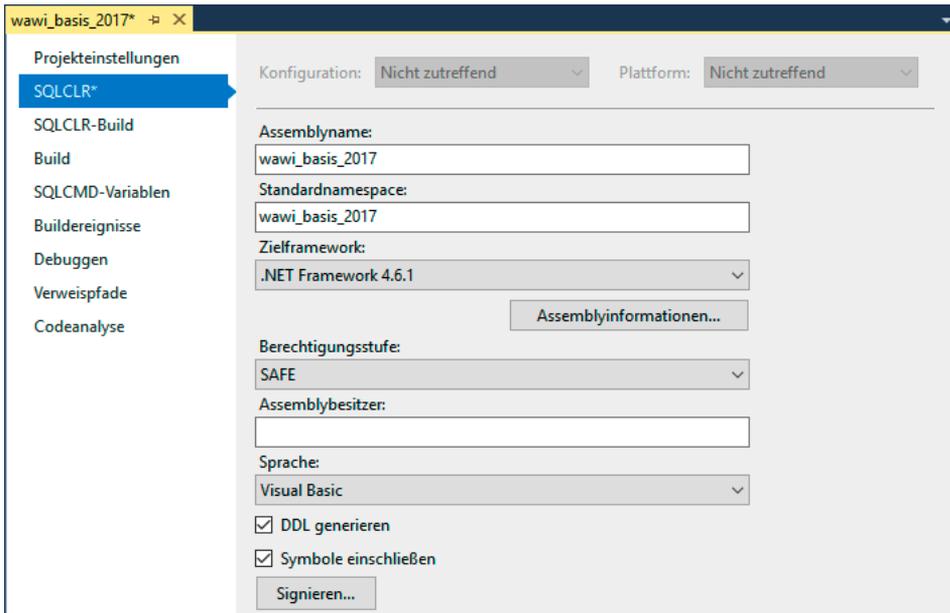


Bild 7.5 Projekteinstellungen

Die Zielplattform wird auf der Seite *Projektinstellungen* ausgewählt. Dies kann ein SQL Server ab der Version 2005 oder Azure SQL Database sein. Die Einstellung passt aber nicht das verwendete Zielframework mit an, sondern wirkt sich in erster Linie auf das Deployment aus.

Bevor wir mit dem Programmieren für die CLR beginnen, beschäftigen wir uns mit dem Schaffen der nötigen Voraussetzungen auf dem SQL Server.

■ 7.2 CLR-Aktivierung

Nach der Installation ist die CLR bei jeder SQL Server 2017-Edition zunächst deaktiviert. Sie müssen daher die CLR auf Ihrem Server aktivieren, bevor Sie die nachfolgenden Beispiele ausführen können.



PRAXISTIPP: Wenn Sie nicht wissen, ob CLR bei Ihrem Server schon aktiviert ist, können Sie dies aus dem Systemkatalog *sys.configurations* auslesen.

Verwenden Sie zum Beispiel folgende Anweisung:

```
SELECT name, value, description
FROM sys.configurations
WHERE name LIKE 'CLR%';
```

Sie erhalten:

name	value	description
clr enabled	0	CLR user code execution enabled in the server
clr strict security	1	CLR strict security enabled in the server



HINWEIS: Mit dem SQL Server 2017 ist eine neue Variante für die Sicherheit von .NET-Code eingeführt worden. Diese hat den Namen *CLR strict security* und sie ist standardmäßig aktiviert. Per Update wird dieses Feature auch nachträglich beim SQL Server 2016 integriert, ist aber per default nicht aktiviert, um die Abwärtskompatibilität zu wahren. Auf dieses neue Feature werden wir etwas später in diesem Kapitel eingehen.

Diese Einstellung ist keine erweiterte Einstellung, kann also ohne vorherige Aktivierung der *Advanced Options* geändert werden.

Für die CLR-Aktivierung führen Sie die Systemprozedur `sp_configure` aus. Mit der Anweisung `RECONFIGURE` setzen Sie die zuvor gemachte Änderung sofort aktiv.

```
EXEC sp_configure 'clr enabled', 1;
GO
```

Ist die CLR-Aktivierung noch nicht erfolgt, erhalten Sie nach der Aktivierung, bevor Sie `RECONFIGURE` ausführen, folgende Meldung:

```
Die Konfigurationsoption 'clr enabled' wurde von 0 in 1 geändert. Führen Sie zum Installieren die RECONFIGURE-Anweisung aus.
```

Prüfen wir nochmals die Einstellung, sehen wir, dass diese Eigenschaft nun den Wert 1 (wahr) aufweist, der verwendete Wert aber noch 0 (falsch) lautet.

```
SELECT name, value, value_in_use
FROM sys.configurations
WHERE name LIKE 'CLR%';
```

liefert:

name	value	value_in_use
clr enabled	1	0
clr strict security	1	1

Da es sich bei dieser Einstellung um einen dynamischen Wert handelt, kann er sofort aktiviert werden. Führen Sie dazu bitte die Anweisung `RECONFIGURE` aus.

```
RECONFIGURE;
GO
```

Danach stimmen *value* und *values_in_use* wieder überein.



HINWEIS: Sie können alle nachfolgenden Beispiele erstellen, auch wenn die CLR noch nicht aktiviert ist. Aber spätestens bevor Sie sie ausführen möchten, muss die CLR-Integration aktiv sein.

Offensichtlich gilt die Aktivierung der CLR nur für benutzerdefinierten Code. Denn die in früheren Kapiteln beschriebenen Datentypen *geography*, *geometry* und *hierarchy_id* sind ja als .NET-Datentypen integriert, funktionieren aber auch, wenn die CLR deaktiviert ist.



ACHTUNG! Aufgrund der neuen *CLR strict security* beim SQL Server 2017 gelten höhere Anforderungen, um CLR-Code ausführen zu können. Da ich der Meinung bin, dass diese Neuerungen besser zu verstehen sind, wenn man sich schon mit CLR-Code befasst hat, möchte ich auf diese Neuerungen und ihre Auswirkungen erst am Ende dieses Kapitels eingehen.

Damit die Beispiele (vorerst) funktionieren können, müssen aber unbedingt folgende Voraussetzungen gegeben sein:

- Der Eigentümer der Datenbank muss Mitglied der Serverrolle *sysadmin* sein.
- Die Eigenschaft *TRUSTWORTHY* muss für die betroffene Datenbank auf *ON* gesetzt werden.

Der Eigentümer einer Datenbank kann auf der Seite *Dateien* des Dialogs *Datenbankeigenschaften* eingesehen und auch eingestellt werden (Bild 7.6).



Bild 7.6 Datenbankbesitzer anzeigen und ändern

Auslesen können Sie den Datenbankbesitzer auch mit dieser Anweisung:

```
SELECT d.name AS db, l.name AS besitzer
FROM sys.databases d
INNER JOIN sys.syslogins l ON d.owner_sid = l.sid
WHERE d.name = 'wawi';
```

Einen neuen Datenbankbesitzer können Sie bei Bedarf mit folgender Anweisung festlegen und ihm gegebenenfalls noch die Mitgliedschaft bei den SQL Server-Administratoren erteilen:

```
ALTER AUTHORIZATION ON DATABASE::wawi TO [softelie\alina];
ALTER SERVER ROLE sysadmin ADD MEMBER [softelie\alina];
```

Meist ist der Grund für den Einsatz einer Client-Server-Datenbank, dass die Anzahl der Benutzer steigt. Auch das Größenwachstum der Datenbank und die zunehmende Wichtigkeit der gespeicherten Daten führen häufig zum Einsatz von Server-Datenbanken. Als Argument hierfür wird oft die Sicherheit der Daten ins Spiel gebracht. Doch Datensicherheit beschränkt sich nicht darauf, dass die Datenbank im Falle eines Ausfalls restlos wiederhergestellt werden kann. Oftmals sind es Benutzer, die erhebliche Schäden verursachen. Dabei sind oft nicht einmal böswillige Absichten der Grund – auch wenn diese nicht außer Acht gelassen werden sollten –, sondern viel öfter ist Fehlbedienung die Ursache für beschädigte Datenbestände.

Um solchen Problemen vorzubeugen, bietet der SQL Server Möglichkeiten, durch gezielte Rechtevergabe den einzelnen Benutzern innerhalb der Datenbank nur diejenigen Möglichkeiten zu eröffnen, die sie für ihre Arbeit benötigen.



HINWEIS: Die in diesem Kapitel beschriebenen Funktionalitäten stehen gleichermaßen bei der Express Edition wie bei den anderen Editionen zur Verfügung.

Sie erfahren im Folgenden, wie das Berechtigungssystem des SQL Servers aufgebaut ist und aus welchen Komponenten es besteht. Sie lernen den Weg von der Anmeldung am Server bis zum Zugriff auf die Daten kennen.

■ 10.1 Authentifizierungsmodi – Anmeldungen und Benutzer

SQL Server unterstützt zwei Authentifizierungsmodi, um einen Zugriff auf den Server zu gewähren:

- Windows-Authentifizierung
- SQL Server-Authentifizierung

Je nach Modus werden dabei Windows-Benutzerkonten oder direkt auf dem SQL Server eingerichtete Konten verwendet.



HINWEIS: Bei SQL Server wird bereits beim Setup festgelegt, ob nur die *Windows-Authentifizierung* oder die *Windows-Authentifizierung und SQL Server-Authentifizierung* verfügbar sein soll. Letztere wird auch als *Gemischter Modus* bezeichnet. Sie können diese Einstellung im Bedarfsfall ändern. Wählen Sie dazu im Objekt-Explorer des Management Studios den gewünschten Server aus. Öffnen Sie im Kontextmenü mittels des Befehls **EIGENSCHAFTEN** den Dialog *Servereigenschaften*. Auf der Seite *Sicherheit* können Sie dann diese Einstellung ändern.

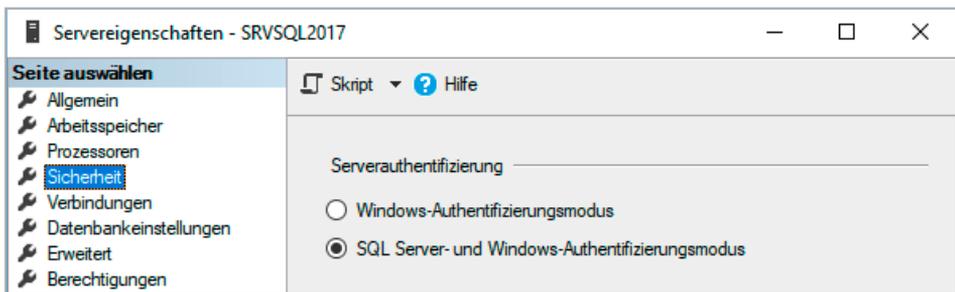


Bild 10.1 Serverauthentifizierung anzeigen und einstellen



ACHTUNG! Beachten Sie, dass bei einer Änderung dieser Einstellung der Serverdienst neu gestartet werden muss. Erst danach ist eine Änderung wirksam.

Diese Einstellung wird in der Registry über die erweiterte Systemprozedur *xp_instance_regwrite* vorgenommen. Der *LoginMode* ist unter dem Registry-Schlüssel *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\INSTANZNAME\MSSQLServer* zu finden. Der Wert 1 steht für die Windows-Authentifizierung; 2 für den gemischten Modus. Der vorletzte Teil des Schlüssels entspricht dem Namen der installierten Instanz. Bei einer SQL Server 2017-Standardinstanz ist zum Beispiel *MSSQL14.MSSQLSERVER* an dieser Position zu finden. Sie können die Einstellung auch direkt hier im Registrierungs-Editor vornehmen. Damit können Sie den Authentifizierungsmodus auf einem Server ändern, wenn einmal kein grafisches Werkzeug zu Verfügung steht. Es kann auch nötig sein, dass Sie den Modus zum Wiedereinrichten eines verlorenen Administratorzugriffs ändern müssen und dies eben mangels Zugriff über die SQL Server Tools nur auf diesem Wege möglich ist. Zu einem diesbezüglichen Beispiel kommen wir noch später in diesem Kapitel.

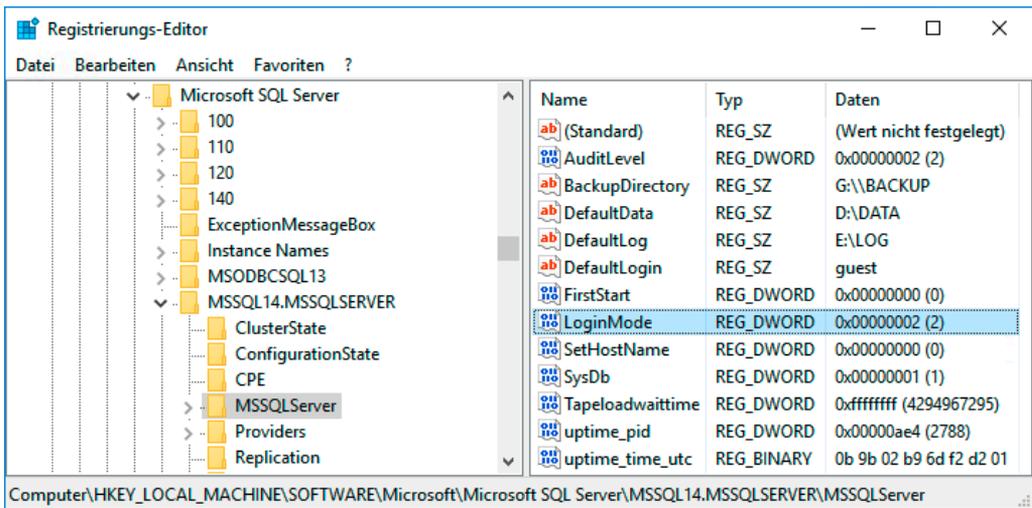


Bild 10.2 Serverauthentifizierung in der Registry

10.1.1 Windows-Authentifizierung

Bei der Windows-Authentifizierung übernimmt der SQL Server den Login von der Domänenanmeldung. Der Anwender muss daher kein separates Kennwort eingeben, wenn er auf den Datenbankserver zugreifen möchte. Die Anmeldung an der Betriebssystemdomäne reicht aus. Dies bedeutet jedoch nicht, dass jeder Benutzer, der sich an der Domäne des Betriebssystems anmelden kann, zugleich auch bereits Zugriff auf den Datenbankserver hat. Der Datenbankadministrator muss einem Betriebssystemkonto explizit das Zugriffsrecht auf den Datenbankserver gewähren.



PRAXISTIPP: Mittels der Windows-Authentifizierung kann nicht nur einem Domänenbenutzerkonto, sondern auch einem Gruppenkonto der Zugriff auf eine Datenbank gewährt werden. Dies kann in einfachen Anwendungsfällen, bei denen keine besondere Differenzierung der Anwender notwendig ist, die Administration vereinfachen. Zugleich ist dies der einzige Fall, bei dem ein Domänenkonto quasi automatisch Zugriffsberechtigungen auf einen SQL Server bekommt. Ist eine Gruppe einmal autorisiert worden, bekommen auch alle später im Active Directory angelegten Konten, welche die Mitgliedschaft in dieser Gruppe erhalten, sofort Zugriff auf den SQL Server.

10.1.2 Gemischter Modus

Der gemischte Modus verwendet sowohl die Windows-Authentifizierung als auch die SQL Server-Authentifizierung. Besitzt ein Anwender aufgrund seiner Betriebssystemanmeldung keine Zugriffsrechte auf den Datenbankserver, kann er sich mithilfe einer SQL Server-Anmeldung anmelden, sofern er eine solche besitzt.



HINWEIS: Clients, die nicht Mitglied der entsprechenden Domäne sind oder sein können, steht nur die SQL Server-Authentifizierung offen. So verwende ich in der Regel SQL Server-Authentifizierung, wenn ich mich von meinem eigenen Notebook auf einem Server bei einem meiner Kunden anmelde.

Dies gilt insbesondere für:

- den Zugriff über einen Webserver,
- andere Betriebssysteme als Windows-Betriebssysteme,
- einen gerouteten Zugriff über ein WAN,
- und auch bei der VNP-Verbindung mittels IPSEC kann in der Regel der Windows-Benutzer nicht verwendet werden.

Sollte einer dieser Fälle bei Ihnen vorliegen, so konfigurieren Sie für Ihren Server bitte unbedingt den gemischten Modus.

10.1.3 Anmeldung und Benutzer

Einer der wichtigsten Punkte beim Sicherheitskonzept des SQL Servers ist die Trennung von Anmeldung und Benutzer:

- **Anmeldung (Login):** Mittels der Anmeldung erhält man Zugriff auf den Datenbankserver. Die Anmeldung erfolgt mit einer der beiden zuvor beschriebenen Authentifizierungsmethoden. Anmeldungen werden auf Serverebene erstellt und daher in der Systemdatenbank *master* gespeichert, ebenso wie Berechtigungen auf Serverebene. Anmeldungen wurden in früheren Versionen auch als *Systembenutzer* und *Sysuser* bezeichnet. Heute ist auch der Begriff *Server Principal* gebräuchlich.
- **Benutzer (User):** Jede Anmeldung benötigt einen zugewiesenen Benutzer in einer Datenbank, um auf diese zugreifen zu können und dort Berechtigungen zu erhalten. Benutzer und deren Berechtigungen werden in der jeweiligen Datenbank gespeichert. In früheren Versionen wurden auch die Bezeichnungen *Datenbankbenutzer* und *User* verwendet. Der Begriff *Database Principal* kommt heutzutage auch zum Einsatz.

Die Trennung von Anmeldung und Benutzer hat vor allem zwei Gründe:

1. Durch die Trennung ist eine sinnvolle Integration der Windows-Authentifizierung in den SQL Server erst möglich.
2. Das Berechtigungssystem innerhalb einer Datenbank ist portabel, da es in der Datenbank selber gespeichert ist. Wird eine Datenbank transferiert, „wandern“ alle Benutzer und

Berechtigungen mit. Am Zielsystem müssen lediglich die Verbindungen zwischen den Anmeldungen und den Benutzern neu hergestellt werden.



HINWEIS: Mit der SQL Server-Version 2012 sind als Option sogenannte CONTAINED DATABASES eingeführt worden. Hier wird dieses zweistufige System durchbrochen, indem auf die Anmeldung verzichtet und ausschließlich ein Benutzer in der Datenbank benötigt wird. Ziel ist es, eine Form der Datenbank zu erhalten, die überhaupt vom umgebenden System unabhängig ist und daher ganz einfach von einem Server auf den anderen transferierbar wird. Daher wird auf alles verzichtet, was sich nicht in der Datenbank selber befindet. Und das sind eben auch Anmeldungen beziehungsweise Logins.

Die nachfolgende Abbildung zeigt das Standardschema eines Zugriffs auf Datenbanken.

- Die Anmeldung erfolgt mittels Windows- oder SQL Server-Authentifizierung.
- Einer Anmeldung können Serverrollen für Berechtigungen auf Serverebene zugewiesen werden.
- Einer Anmeldung können in einer Datenbank Benutzer zugewiesen werden. Erst dadurch erlangt ein angemeldeter Benutzer Zugriff auf die Datenbank.
- Innerhalb einer Datenbank werden den Benutzern in der Regel durch Rollenmitgliedschaften Berechtigungen erteilt.

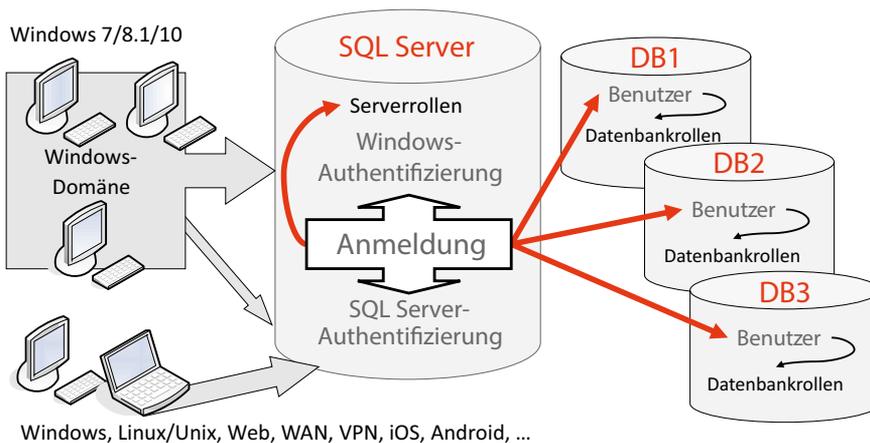


Bild 10.3 Schema eines Datenbankzugriffs

Benutzen Sie eine CONTAINED DATABASE, ändert sich diese Logik. Das Schema zeigt Bild 10.4. Der Zugriff erfolgt direkt auf die Datenbank, ein Zugriff auf andere Serverobjekte ist nicht möglich. Der Server wird quasi wie ein Tunnel passiert und direkt auf eine eigenständige Datenbank – so lautet die deutsche Übersetzung für diese Datenbankart – zugegriffen.

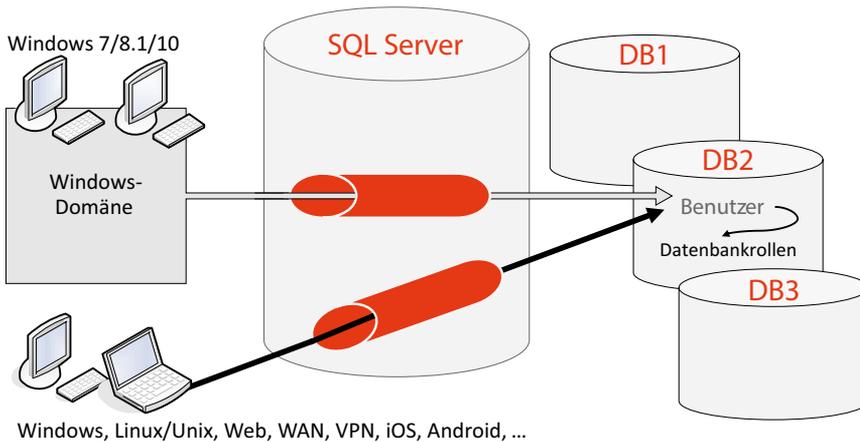


Bild 10.4 Zugriffsschema auf eine eigenständige Datenbank

Die einzelnen Schritte, um Zugriff auf eine Datenbank zu erhalten, werden in den folgenden Abschnitten genauer erläutert.

■ 10.2 Berechtigungen

Jeder Benutzer benötigt Berechtigungen, um innerhalb einer Datenbank etwas tun zu können. Hierbei wird zwischen Objektberechtigungen und Anweisungsberechtigungen unterschieden.

Objektberechtigungen erlauben den Zugriff auf Objekte innerhalb der Datenbank. Diese Objekte sind Tabellen, Spalten einer Tabelle, Sichten, gespeicherte Prozeduren und benutzerdefinierte Funktionen. Für Tabellen und Sichten werden zum Beispiel die Rechte SELECT, INSERT, UPDATE, und DELETE erteilt. Für Tabellen kann auch das Recht REFERENCES (DRI) vergeben werden. Für gespeicherte Prozeduren und benutzerdefinierte Funktionen gibt es das Recht EXECUTE.

Anweisungsberechtigungen werden „gewöhnlichen“ Datenbankbenutzern in der Regel nicht gewährt. Sie beziehen sich nicht auf bestehende Objekte, sondern legen fest, wer Datenbankobjekte erstellen, verwalten und sichern darf. Anweisungsberechtigungen sind beispielsweise CREATE DATABASE, CREATE TABLE, CREATE VIEW und CREATE PROCEDURE. Das Recht BACKUP DATABASE wird benötigt, um eine Sicherung der Datenbank durchführen zu können.

SQL Server 2017 auf Linux

Was vor einigen Jahren noch absolut undenkbar gewesen ist, ist mittlerweile Realität. Microsofts Öffnung hin zu anderen Ökosystemen hat es möglich gemacht, dass nun erstmals ein SQL Server auf einer Nicht-Windows-Plattform zur Verfügung steht. Bereits die Ankündigung seitens Microsoft, den SQL Server auch für Linux auf den Markt zu bringen, hat einer kleinen Revolution geglichen. Da diese Ankündigung knapp vor der Veröffentlichung des *SQL Server 2016* gekommen ist, haben seinerzeit alle angenommen, dass diese Version im darauffolgenden Jahr als Linux-Version nachgeschoben werden würde. Erst ein knappes halbes Jahr nach dem Erscheinen des SQL Server 2016 ist mit den ersten Vorabversionen des *SQL Server vNext*, wie die Bezeichnung der neuen Version vor der Vergabe einer fixen Versionsnummer gelaftet hat, klar geworden, dass es nochmals eine ganz neue Version des SQL Servers werden wird, die nun als *SQL Server 2017* vorliegt.

Implementierung des SQL Servers auf Linux

Beim SQL Server von Linux handelt es sich im Grundprinzip nicht um ein eigenes parallel entwickeltes Produkt, sondern um die Implementierung derselben Datenbank-Engine in einer anderen Umgebung. Dafür verantwortlich ist der sogenannte *SQL Server Platform Abstraction Layer*, kurz *SQLPAL* genannt. Dieser fungiert als Zwischenschicht zwischen dem SQL Server und dem Betriebssystem, die alle Aufrufe zwischen den beiden Systemen abwickelt. SQLPAL fordert zum Beispiel vom Betriebssystem den Speicher an und kümmert sich um das Lesen und Schreiben der Daten von den Festplatten (IO). In Bild 12.1 sehen Sie den schematischen Ablauf dargestellt. In einem isolierten Softwareprozess läuft der SQL Server ident unter Windows und Linux. Dieser sendet seine Windows-Aufrufe an den SQLPAL, der dann ABI-Aufrufe (Application Binary Interface) an die Linux-Gasterweiterungen weiterleitet. SQLPAL ist gleichsam jener Teil, der sich bei der Windows- und der Linux-Version unterscheidet.

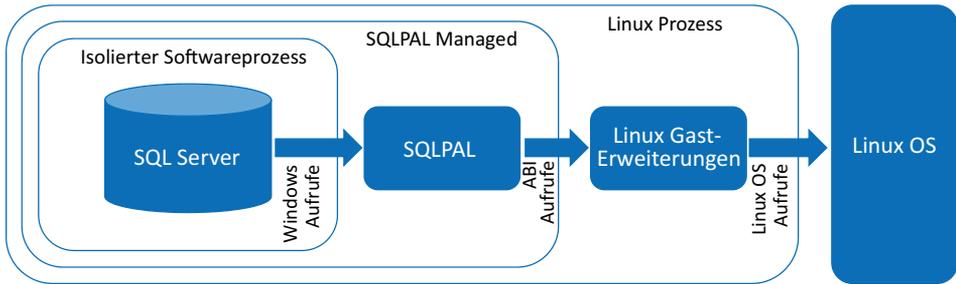


Bild 12.1 Interaktion SQL Server mit Linux über SQLPAL

Unterstützte Plattformen

Im Moment wird der SQL Server für folgende vier Linux-Plattformen unterstützt:

- Ubuntu 16.04 LTS mit EXT4 als Dateisystem.
- Red Hat Enterprise Linux 7.3 oder 7.4 Server, Workstation und Desktop. Als Dateisystem werden XFS und EXT4 unterstützt.
- SUSE Enterprise Linux Server v12 SP2 mit EXT4 als Dateisystem.
- Docker Engine 1.8+ auf Linux, Windows oder MacOS.



HINWEIS: Diese Auflistung entspricht dem momentan aktuellen Stand. Werfen Sie vor der Installation einen Blick auf diese Seite, vielleicht hat es in der Zwischenzeit ja schon eine Aktualisierung oder Erweiterung gegeben:

<https://docs.microsoft.com/de-de/sql/linux/sql-server-linux-release-notes>

Auf dieser Seite finden Sie zusätzlich eine Liste der unter Linux nicht unterstützten Features. Werfen Sie auch dazu einen Blick auf diese Seite, bevor Sie einen Einsatz planen. Von den in diesem Buch behandelten Features werden folgende (zurzeit noch) nicht unter Linux unterstützt:

- Filetable und FileStream (Kapitel 3)
- CLR-Assemblys, die nicht als SAFE (sicher) markiert sind (Kapitel 7)
- Change Data Capture (Kapitel 9)
- Windows-Authentifizierung für Verbindungsserver (Linked Server, Kapitel 10)
- SQL Server-Browser (Kapitel 9)

Als Clientwerkzeuge kommen neben dem *SQL Server Management Studio* (SSMS) ab der Version 17 auch die *SQL Server Data Tools* (SSDT) ab der Version 17 infrage. Auch wenn dies ein wenig verwirrend ist, trägt die Version 17 das Visual Studio 2015, während das Visual Studio 2017 unter der Version 15 geführt wird und damit neuer und natürlich verwendbar ist. Diese zwei Werkzeuge sind nur unter Windows verwendbar, das wird sich in Zukunft auch nicht ändern.

Wenn Sie ein Clientwerkzeug suchen, das auch unter Linux eingesetzt werden kann, kommen für Sie zusätzlich das *Visual Studio Code* mit den Erweiterungen für den SQL Server sowie das *SQL Operations Studio* infrage. Letzteres ist im Moment erst als Vorabversion verfügbar. Diese beiden sind aber für Windows, Linux und MacOS verfügbar.

Visual Studio Code finden Sie hier:

<https://code.visualstudio.com/Download>

SQL Operations Studio finden Sie hier:

<https://docs.microsoft.com/de-de/sql/sql-operations-studio/download>

In dieser automatisch übersetzten Seite wird das Werkzeug als *SQL-Vorgänge Studio* bezeichnet. Ich bezweifle, dass dies der offizielle deutsche Name bei der Veröffentlichung der finalen Version sein wird. Ersetzen Sie in diesem Link de-de durch en-us, gelangen Sie auf die originale englische Seite. Auch wenn ich in diesem deutschsprachigen Buch wenn möglich immer die Links zu deutschen Seiten angebe, verwende ich selber oftmals die originalen englischen Seiten. Dies hilft, wenn die maschinelle Übersetzung der deutschen Seite zu offensichtlich wird und die Gefahr gegeben ist, dass dadurch der Sinn von Aussagen nicht immer klar erkennbar ist.



HINWEIS: Dem *SQL Operations Studio* widme ich mich noch am Ende dieses Kapitels.



ACHTUNG! Begriffe und Vorgehensweisen, die in den vorangegangenen Kapiteln besprochen und erläutert worden sind, werden in diesem Kapitel verwendet, aber nicht nochmals im Detail erläutert. Lesen Sie bitte bei Bedarf die diesbezüglichen Erläuterungen an den entsprechenden Stellen. In diesem Kapitel wird davon ausgegangen, dass die Bestandteile des SQL Servers bekannt sind und Sie zumindest mit den Grundbegriffen von SQL und Zugriffsberechtigungen vertraut sind.

■ 12.1 Installation des SQL Servers

Als eine der am weitesten verbreiteten Distributionen nutze ich Ubuntu für meine Erörterungen in diesem Kapitel. Sie können dazu eine Ubuntu-Maschine in der Azure-Cloud verwenden oder eine lokale Installation vornehmen. Ich installiere für dieses Beispiel Ubuntu in einer virtuellen Maschine auf meinem Server.

Die Installationsdateien für Ubuntu 16.04 LTS finden Sie unter dieser Adresse:

<https://www.ubuntu.com/download/server>

Wenn Sie Ubuntu installiert haben, geht die Installation von SQL Server in wenigen Schritten sehr einfach von der Hand. Der erste Schritt ist, den GPG Key (GNU Privacy Guard) ins

Index

Symbole

\$action 255
\$IDENTITY 176
\$ROWGUID 176
@@ERROR 379
@@FETCH_STATUS 425
(local) 59
@@NESTLEVEL 493, 541
.NET-Code
- Debuggen 638
.NET-Datentypen 563
.NET Framework 562
.NET-Prozedur 389
.NET Stored Procedures 581
.NET-Trigger 596
.NET User-Defined Functions 573
@@ROWCOUNT 327, 412, 434

A

Abbrechen 71
Abbruchbedingung 445
- Trigger 479
Abfrage 219, 233
- mehrere Kriterien 222
- mehrere Tabellen 224
- mit Geodaten 258
- über Zeitraum 755
- Zeitraum 758
Abfrage-Designer 213, 235, 256
Abfrageeditor 69, 256
Abfrageparameter 879
Abgrenzung 5
Ablauf des Kennworts 792
Ablaufdiagramm 336
Ablaufsteuerung 926
Ablaufsteuerungskomponenten 922
Abrunden 307
ABS 307

Abschneiden 514
ABSOLUTE 356
Abstand bestimmen 281
Absteigende Sortierung 242
Abwärtskompatibilität 114
Access 47, 402, 449, 889
- Datenmigration 99
accessadmin 784
Access Database Engine 91
Accumulate 606
ACID 4, 198
Active Directory 980, 983
ActiveX Data Objects 418, 449
ADD MEMBER 813
Ad-hoc-Fehlermeldung 479
Ad-hoc-Sicherung 697
Administrativer Installationspunkt
- SSDT 86
Administratorenkonto 959
Administratorzugriff 826
ADO 449
ADO.NET 151, 390, 418, 461, 562, 581
Advanced Options 568
AFTER 469
Agent 45, 68
Aggregatfunktion 228, 605
Aggregatmethoden 275
Aggregation 248
Aktive Verbindung 683
Aktivieren
- Trigger 499
Aktualisieren
- Datenbank 668
- Datenebenenanwendung 651
Aktueller Benutzer 325
Aktuelles Datum 240
Akzent 33
Alias 79
Aliasname 215, 226

- ALL SERVER 508
- ALTER LOGIN 813, 829
- Alter Natively Compiled Stored Procedure
 - Vorlage 399
- ALTER PROCEDURE 401, 439
- ALTER SEQUENCE 384
- ALTER SERVER ROLE 813
- ALTER TABLE 163, 668
- Always Encrypted 13, 865
 - Konfiguration 867
 - Voraussetzungen 866
- AlwaysOn 16, 67
- Analysezeioptionen 368
- Analysieren 71, 395
- Analysis Services 11, 28, 56
- Anbieter 854
- AND 346
- Ändern 250
 - Kennwort 792
- Änderungen
 - sichern 692
- Änderungsskript 146, 166
- Änderungsweitergabe 139, 140, 154
- Andocken 61
- Anfangsgröße 107, 110
- Anfügen 685
- Anhanggröße 896
- Anlegen
 - Index 144
 - Prozedur 397
 - Schema 796
- Anmeldeinformation 786, 813
- Anmelden
 - eingeschlossene Datenbank 824
- Anmeldename 310
- Anmeldung 778, 810, 965
 - Remote 856
- Anonyme Verlaufstabelle 742
- ANSI_NULLS 368
- ANSI SQL 211
- ANSI-SQL-Anweisungen 390
- ANSI-Standard 354
- Anweisungen skripten 168
- Anweisungsberechtigung 780
- Anwendungseinstellungen 552, 899
- Anwendungsrollen 785, 801
- Anzahl 229
 - Buckets 206
- Application Binary Interface 955
- apt-get 958
- Aqua Data Studio 403
- Arbeitsspeicher 197
- Arbeitsstationsname 325
- ASCII 311
- ASCII-Code 520
- AS EXTERNAL NAME 571
- Assembly 564, 570
 - Berechtigung 591
- Assemblies
 - Sicherheit 619
- Assembly signieren 624, 626
- Assistent
 - Import/Export 55
- Asynchronität 735
- ATOMIC 440, 507, 530
- Atomicity 5
- ATTACH 685, 689
- Attachment 902
- Auditing 735
- Aufgabe 712
 - geplante 712
- Aufgabenplanung 714
- Aufruf
 - Skalarwertfunktion 516
- Aufrufliste 535, 538, 640
- Aufrunden 307
- Aufsplitten 332
- Auftrag
 - manuell starten 919
- Auftrag anlegen 707
- Auftragsfehler 916
- Auftragsverlauf 919
- Ausdruck 158, 168, 226, 232
 - Constraint 135
- Ausdrucks-Generator 939, 940
- Ausfall 692
- Ausführen 71
- Ausführen als 834
- Ausführungsberechtigung
 - Prozedur 398
- Ausführungsberechtigungen 835
- Ausführungsplan 72
 - geschätzter 71
- Ausführungszeioptionen 368
- Ausgabe
 - in Datei 73
 - in Raster 73
 - in Text 73
- Ausgabeparameter 418
- Auslesen
 - Berechtigungen 816
- Auswählen
 - Datenbank 399
- Auswahlkriterien 243
- Auswahlstruktur 336
- Auswertung 6
- Auswertungsmodus 159
- Authentifizierungsmodus 33, 775

AUTO_CLOSE 691
 Automatisches Raster 285
 Automatische Transaktion 360
 Automatische Vergrößerung 107, 111
 Automatisch schließen 691
 Autovervollständigung 966
 AVG 229
 Azure 657

B

Backend
 - Aufgaben 423
 - Programmierung 10
 Backup 692, 703
 - Device 704
 - FILESTREAM 733
 - komprimieren 703
 - Skript generieren 705
 - zeitgesteuert 707
 BACKUP DATABASE 780
 backup_operator 713
 backupoperator 785
 BACPAC 654, 658
 Bandlaufwerk 694
 Basisordner FileTable 189
 bedarfsgesteuert 159
 Bedingung 156, 243
 Bedingungsblock 336
 BEGIN 342
 Beginnt mit ... 245
 BEGIN TRANSACTION 361
 Beispieldatenbank 41
 - generieren 195
 Bemerkung 240
 Benachrichtigungen 917
 Benannte Instanz 30, 768
 Benannte Transaktionen 367
 Benennungsregel 156
 Benennungsschema 125
 Benutzer 778, 814
 - aktueller 325
 - erstellen 796
 Benutzeranzahl 6
 Benutzerbereich 788
 Benutzerdefinierte Aggregate 605
 Benutzerdefinierte Datentypen 562
 Benutzerdefinierte Formate 318
 Benutzerdefinierte Funktion 389
 Benutzerdefinierter Fehler 548
 Benutzerdefinierter Tabellentyp 433
 Benutzerdefinierte Serverrolle 782
 Benutzerdefinierte Variable 290
 Benutzername 795
 Benutzerverwaltung 775
 Benutzerzuordnung 794
 Berechnete Spalte 131, 168
 Berechnung 226, 232
 Berechtigung 397, 398, 780
 - auf Datenbankebene 805
 - auf Serverebene 810
 - auslesen 816
 - effektive 808
 - entziehen 816
 - erteilen 799, 815
 - externe 591
 - für Zeilen 839
 - indirekt 830
 - Verbindungsserver 856
 - vergeben 802
 Berechtigungsarten 800, 803
 Berechtigungsebene 591, 804
 Berechtigungssatz 591, 619
 Berechtigungsstufe 566, 802
 Bereiche 386, 405
 - filtern 245
 Bereitstellen 575, 671
 - Assembly 570
 - DACPAC 649
 Bereitstellungsassistent 652
 Berichtsserver 11
 Beschreibung 130
 Besitzer 110, 789
 - ändern 153
 - Schema 796
 Beständig (Spalte) 170
 Betriebssystemkonto 777
 Betroffene Zeilen 327
 BETWEEN
 - SQL-Operator 245
 Beziehung 153
 - erstellen 137
 bigint 127
 BIGINT 290, 563
 Bildfunktionen 334
 Binärdaten 37, 118
 - speichern 121
 binary 127
 BINARY 290
 Binary Large Objects 118
 Bing Maps 258
 bit 127
 BIT 290
 BLOB 118, 127, 181
 BLOCK-Prädikat 840, 849, 851
 Bogensegmente 259
 Bonne 264
 BREAK 351, 429

- Breite 33
 - Breitengrad 259
 - Browser 771
 - BUCKET_COUNT 206
 - BufferWithCurves 267, 275, 281
 - Build Number 43
 - bulkadmin 781
 - Bulk Copy 932
 - BULK INSERT 555
 - Business Intelligence 10
 - Business Intelligence Development Studio 85
 - ByRef 584
 - ByReference 418
 - Byte 563
- C**
- C# 461
 - Cachewert 383
 - CALLER
 - EXECUTE AS 835
 - Carriage Return 555
 - Cascaded Updates 139
 - CASE 347
 - case sensitive 521
 - CAST 322
 - cat 982
 - Catalog 855
 - CATCH 377, 412
 - cdc-Schema 736
 - CEILING 307
 - CEK 866
 - CERTENCODED() 628
 - CERTPRIVATEKEY() 630
 - CET 764
 - Change Data Capture 735
 - char 126
 - CHAR 274, 289, 311, 563
 - CHARINDEX 311
 - Check 131, 465
 - CHECK-Constraint 200
 - Check-Einschränkung 132
 - CHECK OPTION 851
 - CHECK_POLICY 965
 - CHOOSE 350
 - Circularstring 262
 - Cleanup-Job
 - Change Data Capture 737
 - Client-Anwendungen 449
 - Clientprotokolle 79
 - Clientseitige Entschlüsselung 865
 - Clientseitiger Cursor 452
 - Clientstatistiken 73
 - Clienttool
 - Linux und MacOS 103
 - CLOB 127
 - CLR 389, 561, 645, 786
 - Assembly signieren 624
 - benutzerdefinierte Funktionen 573
 - benutzerdefiniertes Aggregat 605
 - Datenzugriff 581
 - gespeicherte Prozeduren 581
 - Integration 47
 - Komponenten verwenden 612
 - CLR-Aktivierung 567
 - CLR-Debuggen
 - aktivieren 641
 - CLR-Integration 948
 - CLR-Sicherheitseinstellungen 619
 - CLR strict security 568, 619
 - CLR Trigger 596
 - Clustered Index 143
 - Code
 - testen 532
 - Wiederverwendbarkeit 391
 - Codepage 94, 556
 - Code-Vorlage 204
 - COLLATE 33, 201
 - Collation 112, 130, 519, 520, 975
 - auf Datenbankebene 113
 - CollectionAggregate 277
 - COL_LENGTH 308
 - COL_NAME 308
 - Column Encryption Key 866
 - Columnstore-Index 143
 - ColumnsUpdated 468, 596
 - Command-Objekt 453
 - CommandText 453, 582
 - COMMIT 359, 429, 440
 - COMMIT TRANSACTION 361
 - Common Language Runtime 389, 561
 - Completion 942
 - CompoundCurve 262
 - COMPUTE 354
 - Computed Column 168
 - Computerkonto 982
 - Computernamen 325
 - CONCAT 316
 - CONCAT_NULL_YIELDS_NULL 370
 - CONCAT_WS 316
 - Concurrent User 6
 - Configuration Manager 78, 771
 - Connection 582, 929
 - ConnectionString 635
 - Connect-String 452
 - Consistency 5
 - Constraint 131, 465, 479
 - ohne Prüfung aktivieren 135

- Verletzung 376
- Contained Database 114, 818
 - aktivieren 819
 - anmelden 824
- CONTAINED DATABASES 779
- CONTAINED IN 759
- Container 938
- CONTAINS FILESTREAM 123
- CONTINUE 351
- Control Flow 922
- CONVERT 244
- ConvexHullAggregate 276
- COPY_ONLY 698
- COUNT 229, 340
- Crash 729
- CREATE
 - ASSEMBLY 570
 - SEQUENCE 383
- CREATE CERTIFICATE 625
- CREATE DATABASE 105, 117, 689
 - Contained 820
- CREATE FUNCTION 845
- CREATE FUNCTION von UDF geliefert 515
- CREATE LOGIN 812
- CREATE PROCEDURE 392, 439, 571
- CREATE SCHEMA 814
- CREATE SECURITY POLICY 846
- CREATE TABLE 664
 - speicheroptimierte Tabelle 200
- CREATE TRIGGER 288
- CREATE TYPE 297
- CREATE VIEW 235, 248
- CREDENTIAL 786, 813
 - erstellen 813
- CROSS JOIN 268
- CSV 991
 - Import 94
 - CSV-Datei 94
 - importieren 555
- Cumulative Update 43
- Currency 319
- CURRENT ROW 387
- CURRENT_TIMESTAMP 304
- CURRENT_USER 325, 406
- Cursor 353, 525
 - Beispiel 425
 - clientseitiger 452
 - definieren 353
 - Funktionen 301
 - öffnen 356
 - schließen 358
 - schreibgeschützt 355
 - serverseitiger 420
 - Syntaxvarianten 354

- Zeilen abrufen 356
- CURSOR DEALLOCATE 358
- CURSOR FETCH 356
- CURSOR_ROWS 356
- CurvePolygon 263

D

- DAC 645, 649
 - Aktualisieren 651
- DACPAC 600, 646, 654, 667
- DAC-Paketdatei 646
- DacUnpack 657
- DAO 449, 458
- Data Access Objects 449
- Database Engine
 - Access Redistributable 91
- DatabaseMailUserRole 895
- Database Owner 781
- Database Principal 778
- Database Tuning Advisor 83
- Data Control Language 211, 803, 812
- Data Definition Language 163, 211
- Data Flow 922
- Data Layer 2
- Data Manipulation Language 211, 239, 250
- Data Migration Assistant 21
- Data Quality Client 28
- Data Quality Services 11, 27
- Data Query Language 211, 239
- datareader 785
- DataReader 462
- Datasource 860
- Data Source Name 456, 861, 888
- Data Tier Application 645
- Data Tools 88, 561
- Data-Warehouse 11
- datawriter 785
- DATEADD 301
- DATEDIFF 302
- DATEDIFF() 248
- DATEFIRST 371
- DATEFORMAT 372
- Dateianhang 902
- Dateiausgabe 73
- Dateierweiterung 896
- Dateigruppe 106, 107, 161
 - FileStream 174
- Dateigruppensicherung 106, 693
- Datei-Methoden 598
- Dateiname
 - physischer 107
- Dateisystem 118
- Dateivergrößerung 111

- Daten
 - als Mail versenden 903
 - erfassen 149
 - externe 589
- Datenabgleich 251, 923
- DATENAME 303, 348
- Datenänderung 465, 735
 - visualisieren 741
- Datenbank
 - Administrator 777
 - anfügen 681
 - anlegen mit grafischem Tool 109
 - auswählen 399
 - Benutzer hinzufügen 796
 - Besitzer 110
 - Contained erstellen 820
 - Dienste 27
 - differenziell 692
 - erstellen 105, 781
 - importieren in SSDT 614
 - Name 308
 - Objekte 45, 799
 - offline schalten 209
 - Rollen 781, 784, 801
 - sichern 692
 - Snapshot 66
 - trennen 681
 - Trigger 66
 - vollständige Sicherung 692
 - wiederherstellen 715
- Datenbank aktualisieren 668
- Datenbankanwendung
 - Komponenten 7
- Datenbankbenutzer 778
- Datenbankberechtigungsarten 803
- Datenbankdateien 106
- Datenbank deaktivieren 918
- Datenbankdiagramm 46, 137, 152
- Datenbankebene 508
- Datenbankeigenschaften 491, 691
- Datenbank-E-Mail 67, 891
 - einrichten 892
 - konfigurieren 892
 - Standardprofil 895
 - Wiederholungsversuche 896
- Datenbankmodul 56
- Datenbankmodulkonfiguration 33
- Datenbank-Momentaufnahmen 66
- Datenbankname 688
- Datenbankobjekte 45
 - skripten 171
- Datenbankoptimierungsratgeber 39, 55, 83
- Datenbankprogrammierung 10
 - serverseitige 390
- Datenbankprojekt erstellen 660
- Datenbankreplikation 10
- Datenbankrolle 781, 837
 - benutzerdefinierte 784
 - feste 784
- Datenbanktrigger 46
- Datenbank vergleichen 670
- Datenbankverweis 601
- Datenbereichsspezifikation 161
- Datendatei 687
- Datenebenenanwendung 600, 645
 - entfernen 654
 - exportieren 654
 - extrahieren 646, 652
 - Importieren 657
 - registrieren 651
- Datenexport 654
- Datenfeld 289
- Datenflusskomponenten 922
- Datenflushtask 922, 925
- Datenflussziel 935
- Datenimport 89
- Datenimport und -export 40
- Datenlinkdatei 450
- Datenmenge 6
- Datenmigration 99
- Datenquellen-Administrator 456
- Datensammlung 67
- Datensatz
 - ändern 250
 - einfügen 250
 - limitieren 375
 - löschen 250
- Datenschichtanwendung
 - extrahieren 646
- Datenschutzgrundverordnung 7, 865
- Datensicherheit 6, 775
- Datentyp 96, 129, 151, 289, 563, 578
 - geography 258
 - geometry 258
 - konvertieren 322
 - XML 509
- Daten verschlüsseln 865
- Datenverteilung 117
- Datenverzeichnisse 35, 975
- Datenzugriff
 - CLR 581
- Daten zurücksetzen 762
- DATEPART 303
- DATEPART() 406
- datetime 126
- DATETIME 290
- datetimeoffset 126, 187
- DATETIMEOFFSET 305

Datumsformat 373
 Datumskonvertierung 244
 Dauerhaftigkeit 5
 DAY 303
 DB2 4
 dbcli 966
 dbcreator 781
 DB_ID 308
 DB_NAME 308
 db_owner 784, 837
 DCL 211, 803
 DDL 163, 211, 523
 - protokollieren 509
 - unterbinden 509
 ddladmin 785
 DDL_DATABASE_LEVEL_EVENTS 509
 DDL-Trigger 465, 508
 Deaktivieren 918
 - Trigger 499
 DEALLOCATE 358, 426
 Debuggen 60, 71, 532, 638
 - Funktionen 542
 - gespeicherte Prozedur 534
 - Trigger 539
 - Voraussetzungen 532
 Debug-Informationen 417
 decimal 127
 DECIMAL 290, 563
 DECLARE 291, 406
 - CURSOR 353
 DEFAULT 131, 409, 415, 745
 Default-Wert 408
 Deklarieren
 - Variable 291
 DELETE 250, 477
 deleted 255, 471, 596
 DELETE-Trigger 465, 475
 DENSE_RANK 331
 DENY 803, 815
 denydatareader 785
 denydatawriter 785
 Deployment 88, 646, 671
 - SSIS 949
 Disaster Recovery 47, 725
 DESC 242
 Designer 256
 Desktop-Datenbanksysteme 2
 Detach 682
 Details zum Objekt-Explorer 59
 Deterministische Verschlüsselung 874
 Developer Edition 15
 Diagramm 152
 Diagrammbereich 213
 Dienste 78

Dienstkonten 31
 Dienstprinzipalnamen 985
 Dienst starten 972
 Differenzielle Sicherung 692
 - T-SQL 727
 Directory 590
 Direkter Datenbankzugriff 779
 Direkte Rekursion 490
 Direktfenster 535
 diskadmin 781
 DISTINCT 232, 609, 611
 Distributed Management Objects 562
 Distributed Transaction Coordinator 67
 DLL 438, 572
 DML 211, 250
 DML-Trigger 465
 DMO 562, 632
 DNS 982
 Docker Engine 956
 Dokumentieren 240
 Domain-Controller 981
 Domäne 790
 - beitreten 982
 Domänenanmeldung 777
 Domänenkonto 32
 doomed 378
 Doppelte Werte 232
 Double 563
 dpkg 960
 DQL 211, 240
 DROP ASSEMBLY 572
 DROP FUNCTION 517
 DROP LOGIN 813
 DROP MEMBER 813
 DROP PROCEDURE 439
 DROP TRIGGER 467
 DSGVO 7, 865
 DSN 456, 861
 DTSX-Dateien 946
 Duplikate 232
 Durability 5
 Durchlauf 351
 Durchschnitt 229
 DYNAMIC 356
 dynamisches SQL 557
 Dynamische TCP-Ports 772

E

Editieren von Daten 149
 Edition auswählen
 - Linux 958
 Editionen 12
 Editionsaktualisierung 23

- Editor 69, 256
 - Editor nano 981
 - Effektive Berechtigungen 808
 - Eigenschaftsfenster 60
 - Eigenständige Datenbank 779, 818
 - aktivieren 819
 - Eigentum 977
 - Eigentümer 153
 - Eindeutiger Schlüssel 131, 141, 145
 - Einfügen
 - Datensatz 250
 - Eingabemöglichkeiten 129
 - Eingabeparameter 406
 - Eingabeprüfung 479, 506
 - Eingehende Regel 772
 - Eingeschlossene Datenbank
 - anmelden 824
 - Eingeschlossene Spalten 84
 - Einschließendes Oder 246
 - Einschlusstyp 114, 779, 820
 - Einschränkung 131, 243, 465
 - ohne Prüfung aktivieren 135
 - Einstellungen
 - für eigene Anwendung 552
 - Einstellungstabelle 899
 - Einzelbenutzermodus 828, 918
 - Einzelschritt 536
 - Einzelschrittmodus 535, 639
 - ELSE 342
 - ELSEIF 344
 - E-Mail 891
 - bei Fehler 551
 - prüfen 579
 - Versand aus Anwendung heraus 899
 - Warnung 914
 - Empfehlungen
 - Index 84
 - ENABLE TRIGGER 499
 - ENCRYPTION 469
 - Endlosschleife 351, 352
 - Endpunkte 66
 - Enterprise Edition 15
 - Entfernen 250
 - Enthaltene Spalten 147
 - Entpacken
 - Paket 649
 - Entscheidungsstruktur 336
 - Entscheidungsszenarien 5
 - Entschlüsselung 865
 - Entziehen einer Berechtigung 816
 - EnvelopeAggregate 276
 - Equirectangular 264
 - Ereignisprozedur 464
 - Ergebnis
 - tabellarisches 585
 - Ergebnisbereich 217
 - Error-Handling 376, 412, 436
 - ERROR_LINE 379
 - Error-Log 549
 - ERROR_MESSAGE 379, 544
 - ERROR_NUMBER 379
 - ERROR_PROCEDURE 379
 - ERROR_SEVERITY 379
 - Ersatzhandlung 487
 - Erstellen
 - eigenständige Datenbank 820
 - FileTable 183
 - Index 144
 - Serverrolle 783
 - Erste Zeilen 228
 - Erteilen
 - Berechtigung 802
 - Erweiterte IF-Anweisungen 342
 - Evaluation Edition 25
 - EVENTDATA 509
 - Event-Handler 464
 - Ex-aequo-Werte 331
 - Excel 449, 858
 - Excel-Ausgabe 991
 - Excel Export 89
 - EXECUTE 397, 404, 408, 439, 558
 - EXECUTE AS 834, 843
 - execute_as_principal_id 837
 - Execute-Berechtigung 398
 - ExecuteNonQuery 462, 582
 - EXISTS 340, 377, 439
 - Exklusion 247
 - Explizite Transaktion 360, 361
 - Export 55, 89
 - Zertifikat 872
 - Express 59
 - Express Edition 15, 25, 47
 - tägliche Sicherung 71
 - zeitgesteuerte Sicherung 71
 - Expression 232
 - Express mit Advanced Services 48
 - EXT4 956
 - Extended Events 68
 - EXTERNAL_ACCESS 591
 - Externe Assembly 612
 - Externer Datenzugriff 589
 - Extrahieren
 - Datenebenenanwendung 600, 652
- F**
- Facet 156
 - Failure 942

- FAST_FORWARD 355
- Feature Pack 49, 178, 450
- Fehler
 - Benachrichtigung 551
 - erzeugen 380, 381
 - in der Syntax 396
- Fehlerbehandlung 376, 412, 436
- Fehlercode 379, 482
- Fehlermeldung 379, 479
- Fehlerprotokoll 549
- Fehlerursache 544
- Feld 158
- Felddatentyp 289
- Feldeigenschaften 128
- Feldtrennzeichen 556
- Fenster andocken 61
- Fenster-Funktionen 386
- Feste Breite 95
- Feste Datenbankrollen 784
- Festschreiben 359
- FETCH 356, 385, 425, 429
- FETCH_STATUS 358
- FILE 724
- FileInfo 590
- FILELISTONLY 724, 733
- Filestream
 - Zugriffsebene 121
- FILESTREAM 37, 115, 118, 174, 198
 - Recovery 732
- Filestream-Dateigruppe 174
- Filesystem 118
- FileTable 119, 174, 181
 - Basisordner 189
 - erstellen 183
 - Konfiguration 182
 - Ordner erstellen 193
 - Ordnerstruktur 186
 - Ordner umbenennen 190
 - SQL-Zugriff 188
 - Struktur 184
- FILETABLE_COLLATE_FILENAME 184
- FILETABLE_DIRECTORY 184
- FileTableRootpath 189
- FILTER
 - Sicherheitsprädikat 847
- Filterkriterium 232
- Filtern 243
- FILTER-Prädikat 840, 849
- Fingerabdruck 626
- Firewall 26, 40, 57, 532, 639, 770, 862
- FIRST 356
- Flatfile 94
- Flatfilequelle 931, 932
- Flexibilität 6
- float 127
- FLOAT 290
- FLOOR 307
- Flussdiagramm 336
- FOLLOWING 387
- FOR ATTACH 689, 690
- FOR BROWSE 354
- For Each 590
- Foreign Key 131, 135, 137, 465
- FORMAT 317, 319
- Formatierte Mail 905
- FOR-NEXT-Schleife 352
- FOR SYSTEM_TIME 757
- Fortlaufende Nummer 325
- FOR UPDATE 355
- FQDN 985
- Fragmentierung 107
- Framework 564
- Freie Editionen 25
- Freigegebene Funktionen 27, 28
- Fremdprogramme 403
- Fremdschlüssel 131, 135, 137, 154, 200, 665
- Frontend
 - Programmierung 9
- FTP 612
- FTPS 612
- FTP-Task 928
- FULL OUTER JOIN 247
- Full Table Scan 141
- Funktion 46, 299
 - benutzerdefinierte 389, 515, 573
 - Cursorfunktionen 301
 - Debuggen 542
 - Konfigurationsfunktionen 300
 - mathematische 307
 - Metadaten-Funktionen 308
 - Sicherheitsfunktionen 310
 - statistische Systemfunktionen 334
 - Systemfunktionen 322
 - Tabellenwertfunktion 523
 - Zeichenfolge-Funktionen 311
- Funktionen 27
- Funktionskonfigurationsregeln 38
- Funktionsregeln 30

G

- Gebiet 259
- Gemischter Modus 34, 778
- Generieren von Skripten 171
- Generische Skripte 818
- Geodaten 258
 - Flächen 269
 - Index 282

- Methoden 266
- Punkte 268
- Geodatenmodell
 - Typen 259
- Geografieraster 283
- geography 128, 258
- Geometrieraster 283
- geometry 128, 258
- Geschachtelte Prozeduren 424
- Geschachtelter Trigger 490
- Geschachtelte Transaktionen 430
- Geschäftslogik 438
- Geschäftsregel 131, 479
 - prüfen 465
- Gespeicherte Prozedur 46, 390, 399
 - anlegen 393
 - anlegen mit Management Studio 393
 - Aufbau 392
 - debuggen 534
 - indirekter Zugriff 832
 - Input-Parameter 406
 - Transaktionen 428
 - Vorlage 399
- GETANCESTOR 335
- GETDATE 304, 323, 406, 516
- GETDESCENDANT 192, 334
- GetFileNamespacePath 189
- GETLEVEL 335
- GETROOT 334
- GETUTCDATE 304
- Glätten 312
- GLOBAL 355
- Globale Regeln 25
- Globale Variable 291
- Global Positioning System 263
- GO 399, 405
- GPG Key 957
- GPS 263
- Grafische Oberfläche 164
- Grant 802
- GRANT 815
 - EXECUTE 398
- GRANT OPTION 803, 815
- Graph-Daten 2
- Großbuchstaben 236, 312
- Größe
 - Datendateien 107
 - Transaktionsprotokoll 116
- große Objekte 181
- Groß-/Kleinschreibung 520
- GROUP BY 605
- Gruppen 781
- Gruppenbildung 332
- Gruppenfunktion 228

- Gruppenkonto 777
- Gruppierter Index 143, 145
- Gruppierung 216, 248
- Guid 563
- Gültigkeit 579
- Gültigkeitsregel 123, 131, 132
- Gültigkeitszeitraum 747

H

- Haltepunkt 535
- Hash-Index 203, 205
 - Statistik 206
- HEADERONLY 724, 731
- Hierarchie-ID-Funktionen 334
- hierarchyid 128, 187, 192
- Hinweis 240
- History-Table 742
- Hochkomma verwenden 295
- HOST_ID 325
- HOST_NAME 325
- HTML-Format 905

I

- IBinarySerialize-Schnittstelle 609
- IDENT_CURRENT 326
- Identität annehmen 857
- IDENT_INCR 326
- Identität 130
- Identitätswert auslesen 325
- IDENTITY 325
- IDENTITY_INSERT 372
- IDENT_SEED 326
- ifconfig 961
- IF-ELSE 338
- IF UPDATE 476
- IIF 349
- image 127
- IMPLICIT_TRANSACTIONS 373
- Implizite Transaktion 360
- Import 555
 - Datenebenenanwendung 657
 - SSDT 675
- Import/Export-Assistent 55, 89
- Importieren 600
- Imports 574
- IN
 - SQL-Vergleichsoperator 245
- included columns 147
- Index 131, 141, 666
 - eingeschlossene Spalten 84
 - enthaltene Spalten 147
 - erstellen 144

- für speicheroptimierte Tabellen 203
- gruppierter 143
- räumliche Daten 282
- Statistik 142
- zusammengesetzter 142
- Indexempfehlungen 84
- Indextyp 282
- Indirekte Berechtigungen 800
- Indirekte Rekursion 490
- Indirekter Zugriff 830
- INFORMATION_SCHEMA 637, 796, 816
- Init 606
- Inklusion 247
- Inkonsistenz vermeiden 359
- Inline-Funktion 515, 521
- Inline-Tabellenwertfunktion 843
- Inline-View 219
- In-Memory OLTP 197
- INNER JOIN 247
- Input-Parameter 406
- insensitiv 354
- INSERT 250, 477
- inserted 255, 471, 596, 598
- INSERTS generieren 173
- INSERT-Trigger 465, 471
- Installation 18, 24
 - unter Linux 957
- Installationscenter 55, 81
- Installationsregeln 26
- Instanz 767
 - benannte 59
 - Funktionen 27
- Instanzkonfiguration 30
- Instanzname 59, 860
- Instanzstammverzeichnis 30
- INSTEAD OF-Trigger 468, 469, 487
- INSTR 311
- int 127
- INT 290
- Integer
 - Wertebereich 547
- Integration Services 10, 28, 56, 89, 922, 971
 - debuggen 944
 - testen 945
 - Variablen 937
- Integration Services-Katalog 68, 948
- IntelliSense 72, 393, 405, 966
- interfaces 981
- Interne Triggertabelle 604
- IOT 493
- ISDATE 304
- ISDESCENDANTOF 335
- IsInvariantToOrder 607
- IS_MEMBER 310

- IS_MEMBER() 844
- ISNULL 231, 294, 326, 327, 371
 - Beispiel 236
- ISNULL() 497
- ISNULL()-Funktion 236
- Isolation 5
- ISPAC 946, 949

J

- JDBC 50
- Jet-Datenbank-Engine 458
- JOIN 213, 246, 969
- JOIN-Bedingung 843
- JOIN-Klausel 522
- Json 991

K

- Kana 33
- Kartesisches Produkt 268
- Kaskadierende Änderungen 139, 154
- Katalog 855
 - erstellen 948
- Kennwort ändern 829
- Kennwortkomplexität 965
- Kennwortrichtlinie 791
- Kerberos 980
- Kerberos-Ticket 984
- KEYSET 355
- Keytab-Datei 985
- kinit 984
- Klasse 464
- Klausel
 - GROUP BY 248
 - INSERT 250
- Kleinbuchstaben 312
- Kommandozeilentool 76, 793, 880, 962
- Kommentar 240
- Kompatibilitätsgrad 112, 114, 117
- Komplexe Ausdrücke ersetzen 517
- Komplexe Eingabeprüfungen 503
- Komplexität verringern 234
- Komponente
 - Rebex 612
- Komponente in CLR 612
- Konfiguration
 - Kerberos 983
- Konfigurationsfunktionen 300
- Konfigurations-Manager 32, 39, 55, 78, 771, 827
- Konsistenzmodell 4, 198
- Kontrollstruktur 336
- Konvertierung 244, 563
- Konvertierungsfunktionen 322

Koordinaten 265
 Kosten 5
 krb5.conf 983
 Kreisbogensegmente 259
 Kreuzprodukt 268
 Kriterien 216, 243
 Kriterienbereich 215
 ktutil 986
 Kürzeste Entfernung 267
 Kürzeste Verbindung 280

L

Länge 129
 - bestimmen 281
 Längengrad 259
 LANGUAGE 300, 373, 440
 LAST 356
 Latin1_General_CI_AS 184
 Laufende Summe 387
 Laufnummer 130
 LDF 106
 Leer 236, 245
 Leerzeichen 813
 - einfügen 314
 - entfernen 312
 LEFT 311
 LEFT OUTER JOIN 247
 Leistungstools 82, 83
 LEN 311
 Lesezugriff 785
 LIKE 222, 246
 Line 259
 Line Feed 555
 LINESTRING 259
 Linie 259
 Linked Server 66, 852, 854
 - löschen 860
 - Sicherheitseinstellungen 856
 Linux 955
 - Clienttool 56, 103
 - Dienst starten 972
 - Domäne beitreten 982
 - Integration Services 971
 - Kommandozeilentools 962
 - nicht-unterstützte Features 956
 - Server-Agent 970
 - SQL Editor 957
 - Text-Editor 981
 - Umgebungsvariable 964
 - Windows-Authentifizierung 980
 LOCAL 355
 LocalDB 3, 15, 565
 LocalSystem 695

LOCK_TIMEOUT 374
 Login 778, 810, 965
 - Remote 856
 LoginMode 776
 Login-Name 310
 LoginSecure 635
 Logischer Name 107
 Logisches AND 246
 Logisches Nein 246
 Logisches OR 246
 Log-Reader 735
 Log Sequence Number 737
 Lokal 535, 640
 Lokale Variable 291
 Löschen
 - Funktion 517
 - von Datensätzen 250
 Löschweitergabe 139, 140, 154
 LOWER 312
 LTRIM 312, 370
 Luftlinie 269

M

MacOS 957, 966, 988
 - Clienttool 56, 103
 Mailanhang 902
 Mailempfänger 916
 Mail mit Datenanhang 904
 Mail versenden 900
 Mail-Warnungen 914
 MakeCert 625
 Management Studio 39, 54, 56, 109, 393
 - Prozedur ausführen 415
 - Trigger löschen 502
 - Version 58
 Mapbender 269
 MariaDB 4
 Massenimport 932
 master 44
 Master Data File 106
 Master Data Services 11, 28
 Master Encryption Key 866
 - auf Clients verteilen 872
 Mastertabelle 139
 MATCHED 253, 434
 MaxByteSize 607
 Maximalgröße 107
 Maximalwert 383
 Maximum 229
 MDF 106
 Medium 695
 Mehrfachbenutzermodus 919
 MEK 866

Meldung 455
 Memory Optimized Data 115
 Memory Optimized Tables 197
 Mengenoperation 433
 Mercator 264
 MERGE 251, 433, 606
 MessageBox 455
 Metadaten-Funktionen 308
 Methoden
 - Geodaten 266
 Microsoft Management Console
 - Zertifikate 872
 MID 315
 Migration 21
 Migration Assistant 21, 99
 Minimum 229
 Mit Erteilung 803
 Mitgliedschaft 310
 - erteilen 813
 Mitteleuropäische Zeit 764
 Mittelwert 229
 mkdir 977
 model 44
 money 127
 MONEY 290
 MONTH 303
 Mosaikschema 283
 MOVE 724
 MS Access 47, 402, 458, 858, 889
 - Datenmigration 99
 msdb 45
 MS Excel 449
 mssql-cli 962, 966
 mssql-conf 958
 Multiline 262
 Multiple-Row-Funktion 228
 Multipoint 261
 Multipolygon 262
 - Flächen 269
 Mustervergleich 245, 246
 MySQL 4
 - Linked Server einrichten 861

N

Nachfolger 335
 Nachrichtenformat 902
 Nachrichtenversand 891
 Named Instance 59, 768
 Namensauflösung 789, 982
 Namespace 597, 635, 938
 nano
 - Editor 981
 Native Client 90, 450

Natively Compiled Scalar Function 529
 Natively Compiled Stored Procedure 438
 Natively Compiled Trigger 503
 Navigationsschaltflächen 150
 nchar 563
 NDF 106
 Nested Trigger 490
 NESTLEVEL 300, 493, 541
 NET SEND 916
 Netzlast 6
 Netzwerkkonfiguration 79, 771
 Netzwerk Share 182
 Netzwerkverkehr 391
 Neustart 970
 NEWID 176
 NEWSEQUENTIALID 176
 NEXT 356
 NEXT VALUE 383
 Nicht gruppierter Index 203
 NOCOUNT 374
 NONCLUSTERED 201
 Nonclustered Index 143
 Normalisierung 320
 NoSQL 2
 NOT MATCHED 253
 ntext 127
 NTILE 332
 NULL 231, 245, 369
 - eingeben 152
 - ersetzen 231
 - festlegen 140
 - unterdrücken 236
 NULL-Werte
 - ersetzen 326
 - Gefahren 294
 - zulassen 130
 numeric 127
 Nummerierung 382
 Nummernspender 382
 nvarchar 126
 NVARCHAR 289

O

Obergrenze 386
 Oberste 200 Zeilen bearbeiten 214
 OBJECT_ID 309, 493
 OBJECT_NAME 309
 Objekt 799
 Objektberechtigung 780
 Objekte umbenennen 421
 Objekt-Explorer 59, 65, 397
 Objekte zusammenfassen 275
 Objektname 789

ODBC 50, 90, 177, 402, 449, 481, 861, 887
 - Data Source Name 861
 - Linux 962
 - Treiber 455
 ODBC-Treiber 861
 ODER-Kriterien 223
 Öffentliches Profil 895
 offline entwickeln 670
 offset 126
 OFFSET 290, 385, 764
 OLAP 11
 OLE DB 90, 449, 854, 861
 OLTP 116, 197
 Online Analytical Processing 11
 Online Transaction Processing 116, 197
 OPEN
 - Cursor 356
 Open Database Connectivity 449
 Open Geospatial Consortium 259
 OPENQUERY 863
 Operations Studio 56, 957, 988
 Operator 158
 Operatoren 916
 Optimierungsrategebers 55
 OPTIMISTIC 355
 Optionaler Parameter 459
 OR 346
 Oracle 4, 288
 ORDER BY 242
 OUTER JOIN 247
 OUTPUT 253
 OUTPUT-Parameter 410, 418, 449, 581, 584
 OWNER
 - EXECUTE AS 835

P

Pager 916
 Paging 385
 Paket 922
 - ausführen 953
 Paketausführungsprogramm 55, 946
 Paketdatei 649
 Parameter 879
 Parameter für Anwendungen 899
 Partial CClass 578
 Partition 386
 PARTITION BY 387
 Pass-Through-Abfrage 402, 460, 889
 Passwort ändern 829
 path_locator 187
 Performance 197, 448
 Performancesteigerung
 - enthaltene Spalten 148
 Performancevorteile 106
 permanent 170
 PERMISSION_SET 592
 Per Referenz 418, 584
 persisted 170
 Pfadeintrag 964
 Pflichteingabe 130
 Phonetische Suche 314
 PHP 50
 Physischer Dateiname 107
 pip 966
 Pipe 585
 Platform Abstraction Layer 955
 PL/SQL 288
 Point 259
 Policy 156
 PolyBase-Abfragedienst 27
 Polygon 259
 Port 770, 961
 Portabilität 6
 Position 332
 PostgreSQL 4
 PowerShell 625, 985
 PRECEDING 387
 Preis 5
 Presentation Layers 2
 Primäre Datendatei 106
 Primärschlüssel 131, 205
 Primärschlüsselverletzung 377
 PRIMARY 106, 108
 PRIMARY KEY 340, 465
 PRINT 410, 415, 585
 PRIOR 356
 Priorität 346
 Privater Schlüssel 630
 processadmin 781
 ProductVersion 301
 Produkt
 - kartesisches 268
 Produktupdates 26
 Profil 892
 - öffentliches 895
 - privates 895
 Profiler 39, 55, 82
 Profilsicherheit 895
 Program Layer 2
 Programmierbarkeit 394
 Programmierwerkzeuge 561
 Programmversion 300, 301
 Projekt 401
 - konfigurieren 952
 Projektmappe 660
 Projektmappen-Explorer 60, 535
 Properties 60, 661

Protokoll
- Fehler in Programmcode 549
Protokoll angeben 773
Protokolle 67, 79
Protokollfolgennummer 737
Protokollfragmentsicherung 718
Protokollieren
- DDL 509
Protokollierung 5, 493
Provider 450, 854, 860
Prozedur 46, 838
- ausführen 404
- debuggen 534
- für speicheroptimierte Tabellen 438
- geschachtelte 424
- gespeicherte 390, 399, 581
- indirekter Zugriff 832
- Input-Parameter 406
- Parameter 406
- speichern 397
- Standardwerte 408
- Transaktionen 428
- über Kontextmenü starten 414
- Vorlage 399
Prozedur-Header 400
Prüfausdruck 579
Prüfstring 579
Pseudotabelle 471
PTQ-Query 402
public 781, 785
Punkt 259
Python 27, 966

Q

Quellcode 641
QueryDef-Objekt 458
QUOTED_IDENTIFIER 368, 375

R

RAISERROR 379, 479, 482
RAM 197
RAND 308
Randomized Encryption 882
Rang 330
RANGE 387
Rangfolgefunktion 330, 386
RANK 330
Raster 283
Rasterausgabe 73
Räumliche Daten
- Index 282
Räumliche Ergebnisse 260

Räumlicher Index 143
Räumliche Überschneidung 278
Read 582
READ ONLY 354
real 127
REAL 290
realmd 980
Rebex 612
Recht entziehen 816
Rechteeübergabe 775
RECONFIGURE 491, 568, 820
Recordset 418, 452
Recovery 715
Red Hat Enterprise Linux 956
Referenzielle Integrität 5, 139
Referenzvariablen 584
Regel
- aktualisieren 139
- ändern 140
- globale 25
- ohne Prüfung aktivieren 135
Registrieren 62
Registrierte Server 60, 61, 64
Registrierung aufheben 654
Registry 827
Reguläre Datenbereichsspezifikation 161
Regular Expression 579
Reihenfolge 242
Reihenfolge von Triggern 485
Rekursion
- direkte 490
- indirekte 490
Rekursive Trigger 490
RELATIVE 357
Remoteanmeldung 856
Remote-Debuggen 532, 639
Remoteserver 852, 854
Remotezugriff 57
REPLACE 312, 518, 520, 610
REPLICATE 314
Replikation 27, 67
Reporting Services 11, 28, 56
Repository 958
Ressourcen
- externe 786
Ressourcenpool 208
RESTORE 724
RESTORE DATABASE 107
RETURN 392, 410, 480, 523, 574, 578, 582
REVERT 843
REVOKE 802, 815, 816
Richtlinie 156, 646
- Auswertungsmodus 159
- Wirkungsbereich 158

Richtlinienverwaltung 67
 RIGHT 311
 RIGHT OUTER JOIN 247
 RLS 840
 Robinson 264
 ROLLBACK 359, 429, 440, 479, 596
 - für DDL 513
 - TRAN 362
 Rolle 781, 837
 - benutzerdefinierte 805
 Rollenmitgliedschaft 310, 794
 - erteilen 813
 Root-Berechtigung 972
 Root-Berechtigungen 981
 ROUND 307
 ROWCOUNT 327, 375, 412, 434
 ROWGUID 175
 ROWGUIDCOL 177
 Row Level Security 840
 ROW_NUMBER() 332, 386, 756
 ROWS 387
 R Services 27
 RTRIM 312
 Rückgabewert 418
 - von Prozeduren 409

S

sa 57
 sa-Kennwort setzen 976
 Savepoint 364
 Schachtelungstiefe 300, 490, 493
 Schema 788
 - anlegen 796
 - erstellen 814
 - Name 579
 - Objekte 788
 Schemabindung 440
 Schemavergleich 670, 673
 Schleife 525
 Schleifenende 351
 Schließen
 - Cursor 358
 Schlüssellänge 626
 Schlüsselversionsnummer 986
 Schnittmenge
 - Geodaten 267
 Schreibgeschützt 107
 Schreibzugriff 785
 Schrittweite 326
 Schweregrad Fehler 379
 SCOPE_IDENTITY 325
 SCROLL 355
 Securables 799, 807
 securityadmin 781, 785
 Security Policy
 - definieren 846
 SELECT 219, 233, 240
 SELF
 - EXECUTE AS 835
 Self-Join 268
 Semantische Suche 27
 Send 585
 SendResultsRow 585
 SendResultsStart 586
 Sensible Daten 865
 Sequenz 382
 Serialisierung 607
 Server
 - registrieren 62
 - Version 300, 301
 serveradmin 781
 Server-Agent 45, 68, 709, 962
 - Linux 970
 Server-Agent-Auftrag 953
 Serveranmeldung 790
 Serverberechtigungen 810
 Server-Datenbanksysteme 2
 Serverebene 508
 Servereigenschaften 490, 690, 776
 Servergruppen 62
 SERVERNAME 300
 Serverobjekte 66, 694, 854
 Serveroptionen
 - Verbindungsserver 860
 Server Principal 778
 SERVERPROPERTY 301, 974
 Serverprotokoll 40
 Serverrolle 781
 - benutzerdefinierte 782
 - erstellen 783
 Serverrollen 779
 Serverseitige Datenbankprogrammierung 390
 Serverseitiger Cursor 420
 Serversortierung 975
 Service Broker 11
 Servicename 59
 Service Pack 43
 ServicePrincipalName 985
 SET 292
 - ANSI_NULLS 368
 - CONCAT_NULL_YIELDS_NULL 370
 - DATEFIRST 371
 - DATEFORMAT 244, 372
 - Default 140
 - IDENTITY_INSERT 372
 - LANGUAGE 373
 - LOCK_TIMEOUT 374

- NOCOUNT 374
- NOCOUNT ON 408, 472
- NULL 140
- Optionen 368
- QUOTED_IDENTIFIER 375
- RECURSIVE_TRIGGERS 492
- ROWCOUNT 375
- setspn 985
- Setup 18
- SFTP 612, 618
- Share 182
- ShortestLineTo 280, 281
- Sicherheit 6, 66, 775
 - Verbindungsserver 856
 - von Windows NT 451
- Sicherheitseinstellungen
 - Assembly 592
 - CLR 619
- Sicherheitsfunktionen 310, 843, 855
- Sicherheitskontext 858
- Sicherheitsprädikat 847
- Sicherheitsrichtlinie 840
 - ändern 849
 - Berechtigungen 852
 - erstellen 846
- Sichern 681
- Sicherung
 - komprimieren 703
 - tägliche 710
 - vollständige 692
 - zeitgesteuerte 707
- Sicherungsauftrag 707
- Sicherungsfähige Elemente 799
- Sicherungsmedium 66, 694, 704
- Sicherungspfad 704
- Sicherungspunkt 364
- Sicherungssatz 697, 717
 - anzeigen 731
 - auswählen 720
 - Inhalt anzeigen 724
 - überschreiben 704
- Sicherungstyp 698
- Sicherungsvarianten 692
- Sicherungsziele 694
- Sicht 46, 212, 233, 248
 - Abfrage-Designer 235
 - indirekter Zugriff 831
 - mit Parameter 521
- Signaturalgorithmus 624
- Signieren
 - Assembly 626
- Signieren einer Assembly 624
- Single 563
- Skalarfunktionen
 - für systeminterne Tabelle 529
- Skalarwertfunktion 515
- Skalierbarkeit 7
- Skript 166, 573
 - Datei 397
 - für Daten 173
 - generieren 171, 470
 - generisches 818
 - Projekte 401
- smalldatetime 126
- SMALLDATETIME 290, 563
- smallint 127
- SMALLINT 290
- smallmoney 127
- SMALLMONEY 290
- SMO 632
 - API 45
- SMO-Namespace 635
- SMTP 891
- SMTP-Konto 894
- SMTP-Server 893
- Snapshot 66
- Sommerzeit 764
- Sonderzeichen im Namen 813
- Sortierreihenfolge 519
- Sortierung 112, 130, 141, 216, 242, 520
 - auf Datenbankebene 113
- Sortierungskennzeichner 32
- SOUNDEX 314
- sp_addlinkedserver 859
- sp_addlogin 812
- sp_addmessage 380
- sp_addsrvrolemember 813
- sp_add_trusted_assembly 620
- sp_addumpdevice 696, 704
- Spalte
 - berechnete 131, 168
- Spaltenalias 227
- Spaltenberechtigungen 806
- Spalteneigenschaften 128, 129
- Spaltenhauptschlüssel 866
 - auf Clients verteilen 872
- Spaltenname
 - Ändern für Ausgabe 226
- Spaltentrennzeichen 556
- Spaltenüberschrift 215
- Spaltenverschlüsselungsschlüssel 866
- Spaltenzuordnungen 935
- Spatial Reference Identifier 263
- sp_attach_db 689
- sp_configure 491, 568, 820
- sp_detach_db 684
- sp_dropserver 860
- Speichern

- Prozedur 397
- Speichernutzung 208, 980
- Speichernutzungslimit 976
- Speicheroptimierte Tabelle 5, 12, 197
- Speicheroptimierte Tabellen
 - Index 203
- Speicherorte für DB-Dateien 112
- Speicherplatzbedarf 106
- Sperre 353
- SPN 985
- Sprache 959
- Spracheneinstellung 19, 300, 373
- sp_rename 421
- sp_send_dbmail 552, 648, 895, 900
- sp_settriggerorder 485
- SQL 211
- SQL 92-Syntax 354
- SQL Azure 657
- SQL-Bereich 216
- SqlClient 461
- SQLCLR 566
- SQL/CLR-Debugging 641
- SqlCmd 793, 829
- SQLCmd 773, 880, 962
- SQLCMD 40, 50, 55, 76, 712
- SqlCommand 55, 462, 582
- SqlContext 581
- SqlDataReader 582
- SqlDataRecord 585
- SQL-DMO 562
- SQL-Editor
 - Linux 988
- SQLExpress 768
- SOLEXPRESS 31, 59
- SQL Injection 392
- SqlInt32 582
- SQLite 3
- SqlMetaData 585
- SqlMoney 578
- SQL Operations Studio 56, 103, 957, 988
- SQLPAL 955
- SQL Schema Compare 670
- SQL Server 68
 - Agent 707
 - Authentifizierung 34
 - Konfigurations-Manager 771
- SQL Server-Agent 962
- SQL Server-Browser 771
- SQL Server Data Tools 40, 55, 85, 561, 645, 660
 - Business Intelligence 86
- SQL Server-Konfigurations-Manager 55
- SQL Server Management Objects 632
- SQL Server Management Studio 54
- SQL Server Migration Assistant 21
- SQL Server-Objekt-Explorer 402
- SQL Server Profiler 55
- SQL Server-Protokolle 67
- SQL-Skript 171, 196, 401
- SqlString 574
- SQL_VARIANT 127, 290, 517
- SRID 263, 272
- SSDT 55, 85, 561, 660
 - Debuggen 638
 - Transact-SQL-Editor 678
- SSDT-BI 86
- SSIS
 - Ablaufsteuerung 926
 - Datenfluss 932
 - debuggen 944
 - Paket ausführen 953
 - Paket konfigurieren 952
 - Projekt bereitstellen 948
 - Projekte 946
 - Übersichtsbericht 953
 - Variablen 937
- sis-conf 972
- SSISDB 949
- SSIS-Toolbox 925
- SSMA 21, 99
- Stabilität 7, 47
- Standardabweichung 229
- Standarddateigruppe 108
- Standarddatenbank 812
 - ändern 793
- Standarddatenbankrolle 805
- Standarddatenverzeichnis 975
- Standard Edition 15
- Standard festlegen 140
- Standardinstanz 30, 58
- Standardordner 695
- Standardprofil 895
- Standardschema 788
- Standardsprache 792
- Standardwert 131, 408, 409, 745
- Standardzeit 764
- Startreihenfolge
 - Trigger 487, 508
- Startwert 326, 383
- STAsText 266
- Statistik 142
 - aktualisieren 683
- Statistische Systemfunktionen 334
- STDistance 268
- STIntersects 279
- STLenght 281
- STLineFromText 259
- STOPAT 719

Stored Procedure 46, 288, 389
 - anlegen 393
 - debuggen 534
 - indirekter Zugriff 832
 - Ordner 399
 - Transaktionen 428
 STPointFromText 259
 STPolyFromText 259
 STR 315
 Streamen 118
 Streamingzugriff 120
 Streuung 229
 Strict Security 13
 - CLR 568, 619
 String 563
 STRING_AGG() 321, 605
 String-Operationen 578
 STRING_SPLIT-Funktion 526
 Structured Query Language 211
 Struktur einer FileTable 184
 Struktur vergleichen 670
 Subquery 219, 292, 404
 SUBSTRING 315
 Success 942
 Suchbeschleuniger 141
 sudo 958, 972
 SUM 229
 Summary 59
 Summe, laufend 387
 SUSE Enterprise Linux Server 956
 SUSER_NAME() 310, 495, 842
 SUSER_SID() 842
 SWITCHOFFSET 305
 Sybase 288
 Symbolleiste
 - Abfrage-Designer 218
 - Tabellen-Designer 146
 Synchronisieren 251
 Synonym 46
 Syntaxfehler 396, 397
 Syntax Highlighting 966
 Syntaxüberprüfung 71, 218, 395, 579
 sysadmin 310, 781
 sysdac_instances 651
 sysdatabases 308
 SYSDATETIME 232, 304, 323, 406, 516
 SYSDATETIMEOFFSET 304
 syslanguages 373
 sysmail 897
 sysmessages 380, 479
 sysobjects 400
 sys.schema 838
 sys.sql_modules 400
 Systemauswahl 5

Systembenutzer 310, 778, 810
 systemctl 970
 Systemdatenbank 38, 44, 66
 Systemeigen kompilierte Trigger 503
 Systemfunktionen 322
 - statische 334
 Systemintern kompilierte benutzerdefinierte
 Funktion 529
 Systemintern kompilierte gespeicherte Proze-
 dur 438
 System.IO 590, 597
 Systemkonfigurationsprüfung 20
 Systemkonto 695
 Systemprozedur
 - sp_settriggerorder 485
 Systemtabelle 736, 848
 - Datenebenenanwendung 651
 System.Transactions 597
 Systemvariable 291
 Systemversionierung 743
 - entfernen 766
 Systemversionsverwaltung 742
 Systemzeit 240
 SYSUTCDATETIME 304

T

Tabelle 46, 216
 - abgeleitete 219
 - Datenbankprojekte 662
 - erstellen 123
 - temporal 741
 - temporär 555
 - vergleichen 670
 - verknüpfen 246
 - von UDF geliefert 515
 Tabelle für Einstellungen 899
 Tabellen-Aliasnamen 247
 Tabellen-Designer 137, 146
 Tabellentypen 297
 - benutzerdefinierte 297, 433
 Tabellenvariable 298
 Tabellenwertfunktion 515, 523, 843
 - Change Data Capture 737
 TABLE 521
 Table Scan 141
 Table-Valued Parameter 433
 Task 712, 922
 - geplanter 712
 TCO 5
 TCP/IP
 - aktivieren 40
 Teilstring 315
 tempdb 45

- Temporale Tabellen 13, 741
 - Daten ändern 746
 - löschen 748
 - Temporäre Datenbank 36
 - Temporäre Tabelle 555
 - Terminate 606
 - Testen 532
 - text 127, 563
 - Textausgabe 73
 - Textdatei 94
 - Import 94
 - Textlänge 311
 - Textqualifizierer 95
 - Text- und Bildfunktionen
 - statistische Systemfunktionen 334
 - THROW 379, 479, 482
 - time 126
 - TIME 290
 - TIMEOUT 374, 453
 - tinyint 127
 - TINYINT 290
 - Toad 403
 - Tools 53
 - grafische 53
 - TOP 214, 228, 237, 281
 - TOSTRING 334
 - Total Cost of Ownership 5
 - Trace 83
 - Trace-Datei 82
 - TRANSACTION 404, 596
 - Transact-SQL 287
 - Bestandteile 289
 - Cursor 353
 - Datensicherung 703
 - FOR-NEXT-Schleife 350, 352
 - Transact-SQL-Funktionen 515
 - Cursorfunktionen 301
 - Konfigurationsfunktionen 300
 - mathematische Funktionen 307
 - Metadaten-Funktionen 308
 - Sicherheitsfunktionen 310
 - Systemfunktionen 322
 - Zeichenfolge-Funktionen 311
 - Transaktion 5, 6, 47
 - automatische 360
 - beenden 361
 - benannte 367
 - explizit 361
 - geschachtelte 430
 - in Prozeduren 428
 - verteilte 67
 - Transaktions-Log 353
 - Transaktionsprotokoll
 - abschneiden 692
 - sichern 693
 - Sicherung 693
 - Transaktionsprotokolldateien 106
 - Transaktionsprotokollsicherung 721
 - Transaktionssteuerung 211, 359
 - Transformation 922
 - Transparenz 848
 - TreeView 632
 - Trefferquote 142
 - Treiber 50, 450, 854
 - Trennzeichen 95, 556
 - Trigger 66, 288, 389, 464, 596
 - Arten 469
 - für mehrere Ereignisse 477
 - für systeminterne Tabelle 503
 - im Management Studio anlegen 466
 - kombinierte 477
 - löschen 502
 - mit Abbruchbedingung 479
 - rekursiv 490
 - rekursive Aufrufe 539
 - Startreihenfolge 508
 - testen 539
 - Vorlage 466
 - weiterführende Aktionen 470
 - TriggerAction 596
 - TriggerContext 596
 - TRIGGER_NESTLEVEL 493, 495, 540
 - Triggeroptionen
 - eigenständige Datenbank 821
 - Triggerreihenfolge 485
 - Triggertabelle 471
 - TRIM 312
 - TRUNCATE 514
 - Trusted Assembly 623
 - TRUSTWORTHY 590, 612
 - TRY CATCH 377, 412, 436, 582
 - T-SQL 287
 - debuggen 532
 - Typen 297
 - Typumwandlung 315, 940
- ## U
- Übergabeparameter 406
 - Übergabewert 289
 - Überlaufen
 - Transaktions-Log 353
 - Überschneidung
 - Geodaten 267
 - Überschreiben 717
 - Übersichtsbericht 953
 - Überwachen 538
 - Überwachungsausdruck 538

- Ubuntu 956
 - UDA 605
 - UDF 46, 389, 562
 - UDP 772
 - Umbenennen
 - von Objekten 421
 - Umgebende Kurve 267
 - Umlaut 520
 - Umlenken 487
 - UNBOUND PRECEDING 387
 - UNC-Pfad 704
 - Unicode 556, 813
 - UNION 280
 - UnionAggregate 275
 - UNIQUEIDENTIFIER 175, 563
 - UNIQUE KEY 131, 141, 176
 - Universal Time Coordinated 304
 - Unterabfrage 219, 292, 404
 - unterbinden
 - DDL 509
 - Untergrenze 386
 - Unterstützungsobjekte 152
 - UPDATE() 250, 468, 477
 - UPDATE-Trigger 465, 475
 - Upgrade Advisor 21
 - Upgradeplan 653
 - UPPER 236, 312
 - USE 399, 814
 - User 778, 814
 - User-Defined Aggregates 605
 - User-Defined Datatypes 562
 - User-Defined Function 46, 288, 389, 515
 - User-Defined Table Type 433
 - USER_ID 310
 - USER_NAME 310
 - USER_NAME() 842
 - USER_SID() 842
 - USING 253
 - UTC 764
 - UTF-8 556
- V**
- Validierung 479, 506
 - Values-Klausel 435
 - VARBINARY 290
 - varbinary(max) 118, 127
 - varchar 126, 563
 - VARCHAR 289
 - VARCHAR(MAX) 494
 - Variable 289, 563
 - benutzerdefinierte 290
 - deklarieren 291
 - globale 291
 - lokale 291
 - Wertzuweisung 292
 - Variableninhalte
 - anzeigen 537
 - variant 517
 - VBA 449, 454
 - VB.NET 461
 - Verarbeitungsschritt 359
 - Verbinden 70
 - Verbindung 452
 - ändern 71
 - löschen 683
 - Verbindungs-Manager 930, 946, 952
 - Verbindungsparameter 877
 - Verbindungsserver 66, 852, 854
 - löschen 860
 - Sicherheitseinstellungen 856
 - zu anderen DBMS 861
 - Verbindungszeichenfolge 456
 - Verfügbarkeit 7, 67
 - Vergleichsausdruck 232
 - Vergrößerung
 - automatische 107
 - Verketteten 370
 - Verknüpfung 213, 246
 - Verlauf 919
 - Verlaufstabelle 742, 750
 - Veröffentlichen 575, 667, 671
 - Versandstatus 898
 - Verschlüsselung 469, 809, 865
 - Version 960
 - VERSION 300, 301
 - Versionierung 646
 - Versionsnummer 43, 652
 - Schlüssel 986
 - Verteilung 229
 - von Daten 117
 - Vertrauenswürdige Assembly 620
 - Vertraute Verbindung 34
 - Verwalten
 - Sicherheit 784, 790, 801
 - Verwaltung
 - Sicherungsziele 694
 - Verwaltungstools 39, 54
 - Verweigern 803
 - Schreibzugriff 785
 - Verweis 453, 598, 661
 - SMO 634
 - Videos speichern 121
 - View
 - indirekter Zugriff 831
 - VIEW 46, 212, 233, 248
 - Abfrage-Designer 235
 - Visual Basic for Applications 449

Visual Basic.NET 461
 Visual Studio 561, 564
 - Debuggen 638
 - SQL Server-Objekt-Explorer 402
 Visual Studio Code 957
 Visual Studio Shell 55, 85
 Voller Betrieb 692
 Vollständig 113
 Volltextspezifikation 131
 Volltextsuche 10, 27
 Vorbelegter Wert 408
 Vorgang
 - zeitgesteuerter 709
 Vorgänger 335
 Vorlage 204
 - Index 204
 - Prozedur 399
 - Trigger 466
 Vorlagen-Explorer 60, 399

W

Wachstum 107
 Wagerücklauf 555
 Warnung 26, 914
 Warnungen 914, 920
 Wartezeit 374
 Wartungspläne 67
 Webapplikation 392
 Web Edition 15
 Weitere Datendateien 106
 Well-Known Text 259
 Weltzeit 304
 Wenn 347, 349
 Wertebereiche 386
 Werteverteilung 229
 Wertzuweisung 292
 wget 958
 WHEN MATCHED 434
 WHERE 232, 250
 WHILE 338, 350, 358, 525
 Wiederherstellen 681
 - Administratorzugriff 826
 - FILESTREAM 733
 Wiederherstellungsmodell 112, 113, 725
 Wiederherstellungsplan 721
 Wiederholen 314
 Wiederholungsstruktur 336
 Wiederverwendbarkeit 517
 Window-Funktionen 386
 Windows-Authentifizierung 777, 790
 - Linux 980
 Windows-Authentifizierungsmodus 34
 Windows Azure 657

Windows-Benutzer erstellen 813
 Windows-Benutzerkonten 776
 Windows-Firewall 40, 532, 770
 Windows-Registry 827
 Windows-Task 712
 Windows-Zertifikatspeicher 868
 Wirkungsbereich 158
 WITH CHECK OPTION 851
 WITH ENCRYPTION 469
 WITH GRANT OPTION 815
 WITH INIT 726
 WITH REPLACE 717
 WKT 259
 Workflow 922
 - verzweigen 942
 Workstation-Name 325
 Wrapper-Prozedur 459

X

XEvent Profiler 68, 82
 XFS 956
 XML 127, 290, 509, 511
 XML-Index 143
 XQuery 509, 510

Y

YEAR 303

Z

Zeichenanzahl 311
 Zeichenfolge-Funktionen 311
 Zeichenkette 315, 370
 Zeile
 - abrufen 356, 425
 - betroffene 327
 Zeilenberechtigungen 840
 Zeilennummer 332, 379, 397
 Zeilentrennzeichen 556
 Zeilenumbruch 274
 Zeilenvorschub 555
 Zeitabweichung 304
 Zeitachse 719
 Zeitplan 709
 Zeiträume abfragen 758
 Zeitzone 304, 764
 Zentraler Verwaltungsserver 65
 Zertifikat
 - erstellen 625
 - Export 872
 - kopieren 627, 630
 - privater Schlüssel 630

- Zertifikatspeicher 868
- Zielplattform 661
- Zufallsverschlüsselung 874
- Zufallswert 308
- Zugriff
 - direkt auf Datenbank 779
- Zugriffsberechtigungen 775
- Zugriffsrechte
 - Verbindungsserver 856
- Zugriffssicherheit 6, 391
- Zurückrollen 359
- Zurücksetzen auf alten Wert 762
- Zusammenfassung 59, 248, 397
- Zusammengesetzter Index 142
- Zusatzaufgaben 503
- Zusätzliche Verbindungsparameter 877
- Zyklus 383