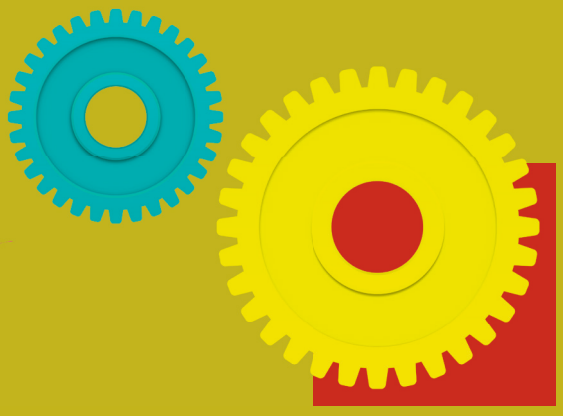
klemens KONOPASEK



SQL SERVER 2017

DER SCHNELLE EINSTIEG

HANSER

Im Internet: Alle Beispiele aus dem Buch

Konopasek

SQL Server 2017

Klemens Konopasek

SQL Server 2017

Der schnelle Einstieg

HANSER

Der Autor:

Klemens Konopasek, Gössendorf/Graz
klemens@konopasek.at

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Lektorat: Sylvia Hasselbach
Copy editing: Walter Saumweber, Ratingen
Umschlagdesign: Marc Müller-Bremer, München, www.rebranding.de
Umschlagrealisation: Stephan Rönigk
Gesamtherstellung: Kösel, Krugzell
Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702
Printed in Germany

Print-ISBN: 978-3-446-44826-1
E-Book-ISBN: 978-3-446-44916-9

Inhalt

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 vonstattengehen. 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 Datensatzebene 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 Gedulcd bedanken.

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

Klemens Konopasek, Gössendorf/Graz

Icons

In diesem Buch werden verschiedene Icons verwendet, deren Bedeutung Sie hier finden.

HINWEIS: Mit diesem Symbol soll auf interessante Informationen besonders hingewiesen werden.

PRAXISTIPP: Mit diesem Symbol sind Informationen gekennzeichnet, mit denen Sie sich das Leben leichter machen können.

ACHTUNG! Sehen Sie dieses Icon, finden Sie Informationen, wie Sie etwas nicht machen oder worauf Sie ein ganz besonderes Augenmerk legen sollten.

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.

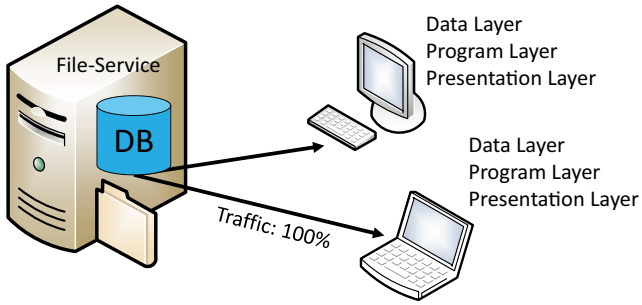


Bild 1.1 Konzept von Desktop-Datenbanken

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.

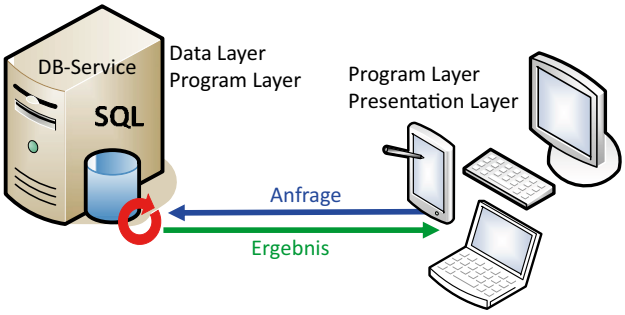


Bild 1.2 Konzept von Server-Datenbanksystemen

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.

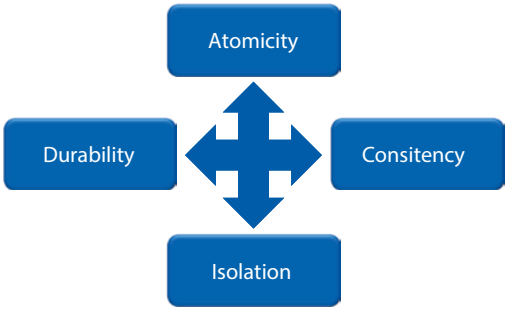


Bild 1.3 Das ACID-Konsistenzmodell

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 Konsistenzerhaltung 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.
- *Netzlast*: 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 (DSGV) 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.

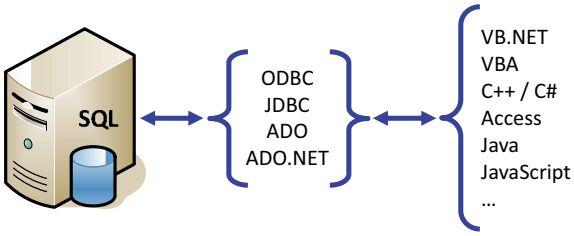


Bild 1.4 Zugriff auf eine Datenbank über Standardschnittstellen

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.

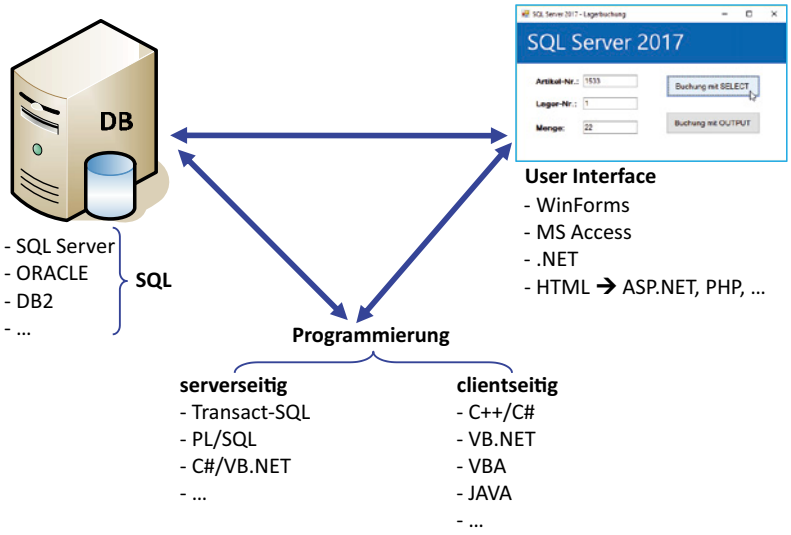


Bild 1.5 Datenbankanwendung und ihre Bestandteile

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.

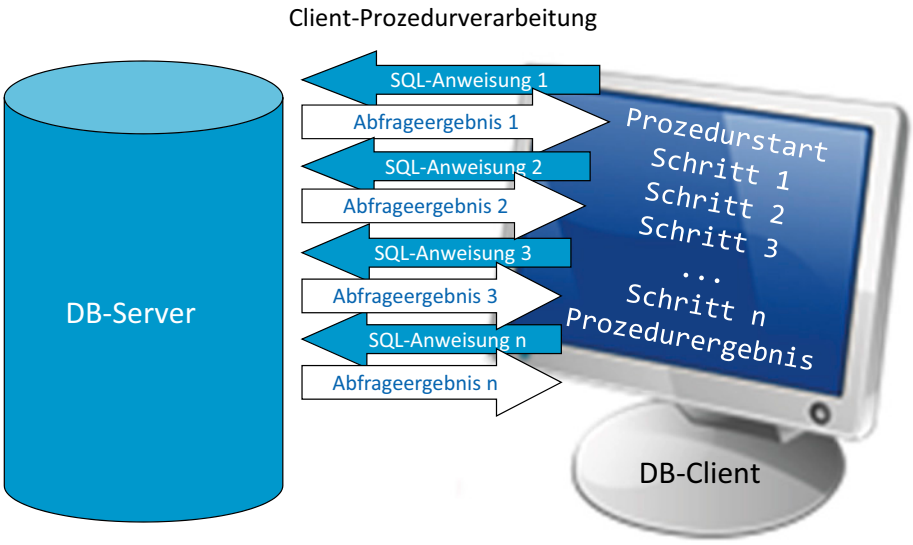


Bild 1.6 Programmlogik im Frontend

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.

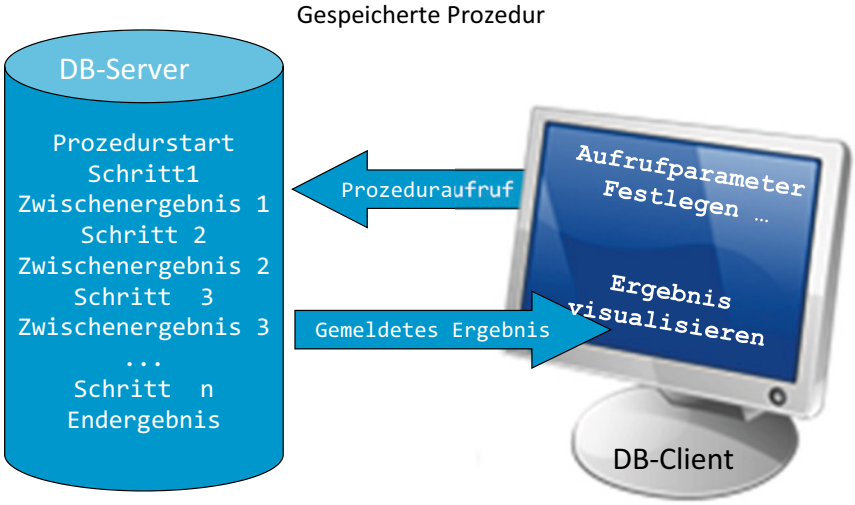


Bild 1.7 Serverseitige Programmierung

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 Datenbankumfeld.

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.