Teil A

SQL Server Grundlagen

In diesem Teil:

Szenarien für SQL Server Entwickler	31
SQL Server Grundlagen	51
Das Arbeitsbeispiel dieses Buches	89

Kapitel 1

Szenarien für SQL Server Entwickler

In diesem Kapitel:

Klassische relationale Datenbanken	32
.NET Inside	34
XML in der Datenbank	36
Verteilte Daten	40
Datenanalyse und Darstellung	42
Immer Online: 7*24 Betrieb mit SQL Server	45
Desktop und mobile Anwendungen	46

Dieses erste Kapitel verfolgt zwei wichtige Ziele: Es soll Ihnen einen Überblick über das »Objekt der Begierde« – SQL Server 2005 vermitteln und wir möchten Ihnen Lust darauf machen, den Server näher kennen zu lernen.

SQL Server 2005 hat mit einem klassischen Datenbankmanagementsystem (DBMS) vor allen Dingen die drei Buchstaben S, Q und L im Namen gemein. Ansonsten bietet er über das effektive Management von relationalen Datenbanken hinaus viele sinnvolle Angebote für Datenbankentwickler. Der Begriff einer Datenmanagementplattform, den Microsoft für diesen Server geprägt hat, ist daher mehr als zutreffend. Wenn Sie die Möglichkeiten kennen, dann können Sie viel eigenen Entwicklungsaufwand sparen und es bringt Sie sicher auf neue Ideen für Ihre Projekte. Anhand einiger Einsatzszenarien stellen wir einige der wichtigsten Highlights von SQL Server vor.

Klassische relationale Datenbanken

Microsoft SQL Server ging mit der Version 4.21 als reinrassiger relationaler Datenbankserver an den Start und auch heute noch bildet die relationale Datenbankmaschine den Kern von SQL Server. Als Entwickler braucht man Sie sicher nicht von den Vorteilen einer Datenbank als Datenspeicher zu überzeugen. Im Gegensatz zu einer dateibasierten Datenhaltung profitieren Sie von bequemen Zugriffsmechanismen und einer an das relationale Datenmodell angepassten Abfragesprache. Die Datenbank-Programmiersprache Transact-SQL ist die Microsoft Implementierung der (mehr oder weniger) standardisierten Datenbanksprache SQL (Structured Query Language).

Warum Transact-SQL heute so ist, wie es ist, hat mit der Geschichte von SQL Server zu tun. Der wurde nämlich in einer Koproduktion von Microsoft, Ashton-Tate und Sybase Ende der 80er Jahre entwickelt. Sybase hatte zu diesem Zeitpunkt schon ein Datenbankmanagementsystem auf dem Markt. Sybase SQL Server 3.0 konnte unter UNIX und VMS betrieben werden. 1989 gab es dann das erste SQL Server für OS/2 (Version 1.0). Das erste Produkt unter dem Namen »Microsoft SQL Server« kam 1992 auf den Markt – immer noch für OS/2. Mit dem Erscheinen von NT 3.1 gab es dann zeitgleich das erste SQL Server für ein Microsoft Betriebssystem: SQL Server 4.21. Die Codebasis war weiterhin die des Sybase Servers. Der Version 4.21 und auch den nachfolgenden Versionen, die von Microsoft entwickelt wurden – 6.0 und 6.5 –, merkte man die Herkunft deutlich an. Die Speicherplatzverwaltung beispielsweise kannte »Medien« fester Größe, ein Überbleibsel aus VMS-Zeiten. Vor allen Dingen aber die Programmiersprache Transact-SQL erinnert an diese Zeit. Transact-SQL wurde ursprünglich von Sybase entworfen und innerhalb von Microsoft SQL Server weiterentwickelt. Im Grunde müsste man von Sybase Transact-SQL und Microsoft Transact-SQL reden. Sybase Entwickler kommen wunderbar mit Microsoft SQL Server zurecht. Schlüsselworte und Syntax der zentralen Sprachelemente sind immer noch identisch – allerdings hat »unser« SQL Server im Laufe der Zeit eine Menge sinnvoller Sprachelemente spendiert bekommen, während Sybase Transact-SQL noch sehr stark an die ursprüngliche Variante erinnert.¹

¹ Wer sich davon überzeugen möchte, für den gibt es in der Link-Sammlung im Dokumentenordner der Buch-CD die Adresse der Sybase-Sprachdefinition.

Falls Sie noch nicht viel mit Microsoft SQL Server zu tun hatten, oder sich noch einmal vergewissern wollen, mit was für einem System Sie da eigentlich die ganze Zeit arbeiten, folgt hier Steckbrief des relationalen Servers:

- **Standardtreue:** Es gibt eine Menge offizieller ANSI-Standards für die Structured Query Language. Trotzdem können sich zwei SQL-Indianer verschiedener Stämme (Datenbanksysteme) nicht miteinander verständigen, jedenfalls nicht, wenn die proprietären Hersteller-Spracherweiterungen eingesetzt werden. Beschränkt man sich bei der Server-Programmierung auf dem ANSI 92 Entry-Level, dann gibt es keine Probleme bei einer Portierung. Alle anderen ANSI-Level und Standards werden nur teilweise eingehalten. Auch wenn Sie vorhaben, eine Datenschicht zu implementieren, die sich auf andere Systeme migrieren lässt, müssen Sie nicht auf die bequemen und mächtigen Spracherweiterungen von T-SQL verzichten. Sie sollten dann allerdings die Geschäftslogik in Prozeduren verpacken, um saubere Schnittstellen zu definieren. Die »Innereien« der Prozeduren – Ihren Programmcode – müssen Sie natürlich konvertieren. Das ist wegen der Einfachheit der Serversprachen aber kein kompliziertes, höchstens ein langwieriges, Unterfangen.
- **Intelligente Optimierung:** Das Wunderbare an der deklarativen Datenbankabfragesprache T-SQL ist, dass Sie auf einem gut erforschten mathematischen Modell beruht: der relationalen Algebra. Ist ein Stapel (Batch) von T-SQL Kommandos erst einmal in eine standardisierte interne Darstellung überführt (algebraisiert), dann beginnt der Optimierer in der Datenbankmaschine mit der Arbeit und findet auf der Grundlage der Datenstrukturen, Tabelleninhalte und Indizes den optimalen Plan für die Abarbeitung der Abfragen. SQL Server verfügt über einen kostenbasierten Optimierer, der sogar die Anzahl der verfügbaren Prozessoren und die Menge an verfügbarem Arbeitsspeicher in Betracht zieht.
- **Spracherweiterungen:** Durch prozedurale Erweiterungen wird T-SQL zu einer Programmiersprache, die dafür geeignet ist, einfache serverseitige Logik zu entwickeln. Das Hauptanwendungsgebiet ist sicher das Schreiben der so genannten gespeicherten Prozeduren. Es gibt einen minimalen, aber ausreichenden, Sprachumfang für die Implementierung der Geschäftslogik. Die Kargheit der Sprache – die mit nur einer Entscheidungs-, einer einzigen Schleifenstruktur und ausschließlich skalaren lokalen Variablen auskommt – wird durch die hohe Ausführungsgeschwindigkeit wettgemacht. Für komplexe serverseitige Aufgabenstellungen bietet sich in SQL Server 2005 die Verwendung von .NET-basierten Datenbankobjekten (wie benutzerdefinierte Datentypen oder gespeicherte Prozeduren) an.

Spracherweiterung gibt es aber auch im relationalen Bereich. In SQL Server 2005 ist es möglich, rekursive Abfragen durch Common Table Expressions zu programmieren. Nummerierte Ergebnismengen werden durch Ranking Functions möglich, Kreuztabellenabfragen durch das Schlüsselwort PIVOT und vieles mehr. Sie sollten sich immer erst nach solch einer T-SQL-Variante umsehen, bevor Sie eine Datenoperation selbst programmieren. So sparen Sie Entwicklungs- und Ausführungszeit.

- **Skalierbarkeit:** Skalierbarkeit hat bei Datenbankmanagementsystemen entweder mit steigenden Benutzer- und Abfragezahlen oder wachsenden Datenmengen zu tun. Antwortzeiten und Durchsatz in einem akzeptablen Bereich zu halten, ist in einem großen Datenbanksystem eine anspruchsvolle Aufgabe. In SQL Server sind verschiedene Komponenten vorhanden, die zur Skalierbarkeit beitragen. Es gibt Caching-Mechanismen für T-SQL-Befehlsstapel, die verhindern,

dass ähnliche Kommandos erneut für die Ausführung vorbereitet und kompiliert werden müssen (*Plan Caching*). Ein Sperren-Manager kümmert sich darum, dass die Sperren auf Datensätze möglichst effizient gesetzt werden (*Lock Manager*). In Systemen mit mehreren Prozessoren können viele der aufwändigen Operationen parallelisiert werden (*Query Parallelism*). Die Liste der Optimierungen ist verdammt lang.

- **Management großer Datenmengen:** Eine Anwendung, die unter Laboranwendungen in der Entwicklungsumgebung ausgezeichnet funktioniert, kann im Echtbetrieb unter der Last großer Datenmengen »einknicken«. SQL Server verfügt dabei über viele Funktionen, die den Umgang mit großen Datenmengen erleichtern. Das beginnt mit der Möglichkeit über Dateigruppen die Objekte in einer Datenbank geschickt zu verteilen und Zugriffe parallel auszuführen, was mit cleveren Indizierungsverfahren zu tun hat. Und es endet bei partitionierten Tabellen, Sichten und Indizes.
- **Autotuning:** Eine der wichtigsten Designvorgaben von SQL Server war von Anfang an, die Entwickler und Administratoren mit so wenig technischen Details wie möglich zu belasten. Keine Frage – andere DBMS sind auch performant, aber es werden viel tiefere Kenntnisse des Systems benötigt, um diese Performance zu erreichen. Die allermeisten Stellschrauben bei SQL Server werden von diesem selbst fein justiert (Autotuning), während andere Datenbanksysteme es beispielsweise erforderlich machen, zwischen verschiedenen Speicherstrukturen auszuwählen. Dennoch ist unbedingt notwendig, sich mit der Arbeitsweise von SQL Server vertraut zu machen. Direkt »aus der Dose« läuft jedes SQL Server zwar schon sehr ordentlich – mit der richtigen Konfiguration und Programmierung kann er aber höllisch schnell werden.

Es gibt eine große Anzahl von Erweiterungen des relationalen Kerns, die Sie natürlich auch in diesem Buch kennen lernen werden. Volltextsuche, XML in der Datenbank, die Möglichkeit, SQL Server Objekte in .NET-Programmiersprachen mit Hilfe des .NET-Frameworks zu implementieren, sind nur ein paar davon. Wie man es auch dreht und wendet: An den relationalen Funktionen führt kein Weg vorbei. Ein Entwickler wird immer gute Transact-SQL-Kenntnisse benötigen, um das Optimum aus seinen SQL Server-Anwendungen herauszuholen.

.NET Inside

Die klassischen relationalen Abfragesprachen für Datenbanksysteme, wie Transact-SQL, weisen drei typische Merkmale auf: Sie sind unglaublich schnell, simpel aufgebaut und mehr oder weniger standardisiert. Die Muttersprache SQL ist hervorragend dafür geeignet effiziente Datenbankabfragen zu liefern. Sie lässt sich besonders leicht analysieren, algebraisieren und optimieren. Die unterliegende relationale Algebra ist vollständig beschrieben und sehr gut erforscht – eine Freude für jede Datenbankmaschine. Auch die Ausführung der kleinen prozeduralen Programmiersprachenanteile in Transact-SQL geschieht mit einer atemberaubenden Geschwindigkeit. Ein kurzer Blick auf den Sprachumfang und die Syntax von T-SQL reicht vollkommen aus, um das zu erklären. Transact-SQL ähnelt mehr einer einfachen Batch- oder Assemblersprache als einer »richtigen« Programmiersprache.

Was für die Datenbankmaschine von Vorteil ist, das ist für Menschen (hier Entwickler genannt) nicht immer schön. Eine große Menge der möglichen Geschäftsprobleme lassen sich zweifellos gut in einem relationalen Schema abbilden. Es gibt aber eine vermutlich fast genau so große Menge, bei der die

Übersetzung etwas holprig wirkt. Da geht es um Themen wie die Abbildung hierarchischer Strukturen beliebiger Tiefe (hier bieten die T-SQL Common Table Expressions aber endlich einen Ausweg). Es geht um die Umsetzung objektorientierter Prinzipien wie Vererbung oder Polymorphie: Ein Kundendatensatz ist ein spezialisierter Adress-Datensatz. Schließlich geht es ganz schlicht und ergreifend um das Fehlen elementarer aber nützlicher Basisdatentypen wie Arrays oder Records (Arrays soll es erst in der nächsten SQL Server-Generation als Datentyp geben).

Betrachtet man das Zusammenspiel zwischen Client- und Serverprogrammierung, dann wird man feststellen, dass ein sehr gut durchdachtes objektorientiertes Design der Geschäftsobjekte einer Anwendung sich häufig nur mit einigem Aufwand in einer Datenbank umsetzen lässt. Das dazu notwendige objektrelationale Mapping kann handgemacht oder automatisiert sein. Es viele gute Tools dafür, die sich sogar in das Visual Studio integrieren lassen. Das Mapping besteht in der Regel aus vielen »dummen« Wrappern um die Tabellen einer Datenbank. Über diese werden die Datenzugriffsmethoden zur Verfügung gestellt. Hier klafft eine Lücke in den Ausdrucksmöglichkeiten der verschiedenen Programmiersprachen, die – entlehnt aus der Elektrotechnik – gerne als Impedanz-Fehlanpassung bezeichnet wird (so – jetzt *klings* das Problem gleich viel besser). Arbeitet man in der .NET-Entwicklung mit typisierten ADO.NET-Datasets anstelle von Geschäftsobjekten, die man von Grund auf selbst implementiert hat, dann stellen sich einige der Probleme nicht mehr so dramatisch dar. Was bleibt ist, je nach Architektur der Anwendung, aber immer noch die Notwendigkeit, serverseitigen Code in T-SQL zu formulieren. Das ist für viele Applikationsentwickler eine ungeliebte Aufgabe – während es für viele Datenbankentwickler eine geliebte Aufgabe ist.

Microsoft hat nicht den Weg anderer Datenbankhersteller beschritten, welche die SQL:1999- und SQL:2003-Standards zur objektrelationalen Erweiterung von SQL umsetzt haben, sondern einen eigenen, leistungsfähigen Weg entwickelt, die Mängel des ursprünglichen SQL auszugleichen: Es handelt sich um die Integration der Common Language Runtime in die Datenbankmaschine. Zu Beginn der Entwicklung von SQL Server 2005 war nur die Möglichkeit vorgesehen, gespeicherte Prozeduren in Visual Basic.Net schreiben zu können. Gott sei Dank haben sich die schlauen Köpfe im Entwicklungsteam das anders überlegt und so ist etwas richtig Gutes daraus geworden. Serverseitiger Code kann in Sprachen wie VB.NET oder C# geschrieben werden, die Möglichkeiten der Base Class Library stehen zur Verfügung, und es lassen sich mit .NET nicht nur gespeicherte Prozeduren implementieren, sondern insgesamt diese Datenbankobjekte:

- Gespeicherte Prozeduren
- Trigger
- Benutzerdefinierte Funktionen
- Benutzerdefinierte Datentypen
- Aggregatfunktionen

Das bietet phantastische neue Möglichkeiten für Entwickler. Allerdings darf man dabei nicht übersehen, dass für den Datenzugriff weiterhin T-SQL eingesetzt wird, und zwar über einen serverseitigen, im Prozessraum der Datenbankmaschine gehosteten Datenprovider. Das Implementieren komplexer Geschäftslogik in der Datenbank selbst wird durch die .NET-Programmierung deutlich komfortabler. Die Ablösung von Prozeduren, die keinen oder nur einfachen prozeduralen Code beinhalten, lohnt sich dagegen nicht. Hier macht T-SQL weiterhin einen guten und schnellen Job. Betrachten Sie »NET-

Inside« als einen zusätzlichen Weg, den Sie in Abhängigkeit von der Gesamtarchitektur Ihrer Anwendung gewinnbringend einsetzen können.

Die Impedanz-Fehlanpassung ist damit jedoch noch nicht gelöst. Da warten wir hoffnungsvoll auf »LINQ-Inside«.²

XML in der Datenbank

Die Speicherung von Daten im XML-Format und die Speicherung von Daten im Format eines relationalen Datenbankmanagementsystems könnten unterschiedlicher nicht sein. XML-Textdokumente sind selbst beschreibend, »geschwätzig« und nehmen keine Rücksicht auf den Speicherplatz. Daten in Datenbankdateien sind binär codiert, »unverständlich« und auf optimale Speicherplatzausnutzung und schnellen Zugriff hin programmiert. Dieser Vergleich soll keine Wertung darstellen! Schließlich sind die Entwicklungsziele der beiden Technologien ursprünglich sehr verschieden gewesen. Die *eXtensible Markup Language* wurde aus der Metasprache *Standard Generalized Markup Language* (SGML) heraus entwickelt, um einen allgemeinen Standard für menschen- und maschinenlesbare Dokument zu schaffen. Man kann als Mitglied der hart arbeitenden IT-Arbeiterklasse gar nicht dankbar genug dafür sein, dass es die Entwicklung hin zu XML und den vielen davon abstammenden Technologien, wie XML-Schemata (XSD), Extensible Stylesheet Language Transformationen (XSLT), XQuery, XPath, XPointer, und so weiter und so weiter und last, not least die XML-basierten Webservices gibt. XML-Technologien werden immer mehr zum Mittel der Wahl, um Interoperabilität zwischen Systemen herzustellen.

XML ist optimal für die Speicherung generischer, aber nichtsdestotrotz klar und eindeutig beschreibbarer Dokumente geeignet und für den Austausch dieser Dokumente zwischen Organisationen und deren EDV-Systeme. Das Datenformat der SQL Server Datendateien dagegen ist optimal für das effektive Speichern »konventioneller« Geschäftsdaten geeignet. Nun ist es aber in der Praxis so, dass die in XML-Dokumenten enthaltenen Daten ebenfalls effizient gespeichert werden müssen, und dass die in SQL Server Datendateien vergrabenen Informationen mit anderen Systemen ausgetauscht werden sollen. Mit dem Abspeichern hunderter von XML-Dokumenten in einem Verzeichnis eines Dateiservers und dem Durchsuchen jedes einzelnen Dokumentes mit der entsprechenden XQuery-Abfrage wird man nicht richtig glücklich werden, was die Effizienz angeht. Und mit dem Versenden einer 20 GB großen SQL Server Datenbankdatei einer Axapta-Installation an eine Firma, die ein SAP-System unter ORACLE betreibt, wird man nicht richtig glücklich werden, was den reibungslosen Datenaustausch und die Konvertierung der Daten angeht (ja ich weiß – auf *solche* Ideen wären Sie nie gekommen; das sind wirklich extreme Beispiele). Es liegt nahe, die Vorteile beider Technologien miteinander zu verknüpfen, um das Beste aus beiden Welten zu bekommen.

Es gibt auf dem Feld der reinrassigen XML-Datenbanksysteme ein paar mehr oder weniger experimentelle Systeme und es gibt den Tamino XML Server als ein sehr weit ausgereiftes Produkt. Diese Server speichern ausschließlich XML-Daten und verwenden die diversen XML-Sprachen für das

² LINQ (Language Integrated Query) ist eine geplante Erweiterung des .NET-Frameworks beziehungsweise der .NET-Programmiersprachen. LINQ soll die Formulierung von Abfragen mit Mitteln der Programmiersprache selbst erlauben. Inwieweit LINQ auf der Serverseite eine Rolle spielen wird, ist allerdings noch nicht klar.

Speichern, Abfragen und Manipulieren. Der Haken bei dieser Philosophie ist zweifellos, dass *lupenreine* XML-Applikationen in der wirklichen Welt doch eher noch die Ausnahme darstellen und die klassischen Datenbankmanagementsysteme bei typischen Geschäftsanwendungen immer noch klar die Nase vorn haben, was die Leistungsfähigkeit angeht. Hersteller wie Microsoft und andere beschreiten daher einen sehr attraktiven zweiten Weg: sie bieten relationale Datenbankmanagementsysteme an, die XML-fähig sind. Hat man solch eine heterogene Datenbank zur Verfügung, kann man sich als Entwickler je nach Aufgabenstellung für die eine oder andere Variante entscheiden oder sogar beide Ansätze kombinieren. SQL Server 2005 stellt mit dem Datentyp `xml` die Möglichkeit zur Verfügung, in jeder beliebigen Tabelle neben relationalen Daten auch XML-Dokumente zu speichern.

Ein dritter Weg stellt die Abbildung von XML-Dokumenten auf relationale Tabellen und Beziehungen dar. In SQL Server besteht diese Möglichkeit schon seit der Version 7.0. Die Transact-SQL `OPENXML`-Funktion ist in der Lage, ein XML-Dokument zu parsen und daraus eine virtuelle Tabelle (ein *Rowset*) zu machen, deren Datenzeilen dann in der Datenbank gespeichert werden können. Umgekehrt existiert mit der Klausel `FOR XML`, die in jedem `SELECT`-Abfragekommando eingesetzt werden darf, ein Mechanismus, der aus Tabelleninhalten ein wohlgeformtes XML-Dokument generiert. Das kann in einfachen Szenarien schon vollkommen ausreichend sein, aber das direkte Speichern von XML-Dokumenten in der Datenbank hat diverse Vorteile:

- Die Datenstrukturen sind dynamisch: Daten, die über XML-Dokumente ausgetauscht werden, sind semistrukturiert und die Schemata unterliegen häufig Änderungen. Ein gutes Beispiel dafür sind Artikelkataloge, die von großen Herstellern im XML-Format zur Verfügung gestellt werden. Mit der Einführung neuer Produktlinien werden in der Regel neue Beschreibungsattribute fällig. Das ist für die resultierenden XML-Dokumente kein Problem – bei der Speicherung in einem relationalen System müssten jedoch die Tabellen angepasst werden. Bei Änderungen in der hierarchischen Struktur möglicherweise sogar neue Tabellen und Beziehungen angelegt werden.
- Die Dokumente können geprüft werden: Beim Einfügen in eine XML-Spalte überprüft SQL Server auf jeden Fall die *Wohlgeformtheit* des Dokuments. Sicherheitshalber sollte man gleich erwähnen, dass auch XML-Fragmente zugelassen werden. Dies sind »Teilbäume« eines XML-Dokuments, die kein Wurzel-Element besitzen, ansonsten aber den XML-Regeln genügen. Als Entwickler können Sie zusätzlich festlegen, dass auch die *Validität* der eingefügten Dokumente überprüft werden soll. Dazu werden ein oder mehrere XSD-Schemata in einer Schemasammlung hinterlegt. Bei Operationen auf den XML-Daten kann ein Schema aus einer Sammlung für die Validierung der Ergebnisse verwendet werden.
- Die Dokumente können indiziert werden: Daten in einer Spalte vom Typ `xml` lassen sich mit einem Volltextindex belegen. Dadurch kann der Inhalt der Dokumente erschlossen werden, ohne dass die Namen von Elementen oder Attributen mit indiziert werden. Zur Schaffung möglichst schneller Zugriffspfade für `SELECT`-Abfragen lassen sich darüber hinaus spezielle XML-Indizes einrichten.
- Es können XML-Methoden verwendet werden: Für das Navigieren, Abfragen und Manipulieren von XML-Daten gibt es XML-typische Methoden, wie XQuery. Sie können diese Methoden in `SELECT`-Abfragen einsetzen, um Daten in XML-Spalten zu durchsuchen. Dabei können Sie innerhalb einer Abfrage ohne Probleme auf XML-Daten und nicht-XML-Daten zugleich zugreifen. Mit einem `UPDATE` unter Verwendung der XML-Methoden lassen sich XML-Dokumente direkt in der Daten-

bank ändern, ohne dass sie aus den Datenstrukturen ausgelesen, geändert und neu gespeichert werden müssten.

Wenn Sie es in Ihrer Entwicklerpraxis mit XML-Dokumenten zu tun haben, dann sollten Sie sich auf jeden Fall mit dem vierten Teil dieses Buchs (»XML«) beschäftigen. SQL Server bietet hervorragende Möglichkeiten für den Umgang mit XML-Daten.

Gekoppelte Systeme und SOA

Verlässt man mit seinen Anwendungen das Entwicklungslabor, gelangt man in die unübersichtliche Wirklichkeit der EDV-Systeme, von denen es viele gibt mit denen die eigene Applikation reden muss und viele, die sich für die Daten in der Applikation interessieren.

Weil das alles so wichtig ist, fliegt seit einiger Zeit ein neues »Buzz Word« umher: Service Oriented Architecture. SOA kann man auf die eine oder andere Art definieren, aber im Kern geht es darum, »lose gekoppelte Systeme« interoperabel zu machen. Dabei stellen EDV-Systeme Daten und Logik auf standardisierte Weise im Netzwerk zur Verfügung. Die Implementierung von SOA ist ganz eng mit dem Thema Webservices (und einigen unterstützenden Technologien) verbunden.

Webservices

Reizvoll an Webservices ist ihre plattformunabhängige und standardisierte Form der Kommunikation. Wenn zwischen Systemen nichts anderes mehr geht, dann geht zumindest noch ein Webservice. Welche Technologie zur Implementierung verwendet wird, ist vollkommen transparent, und so kann man Java-Anwendungen unter Unix mit .NET-Anwendungen unter Windows oder mit Host-Anwendungen kommunizieren lassen. Das geht auch wunderbar im Internet oder Extranet. Das Protokoll *SOAP (Simple Object Access Protocol)*, das sich hinter Webservices verbirgt, ist nicht zwangsläufig an HTTP gebunden. SQL Server stellt Webservices allerdings ausschließlich über HTTP-Endpunkte zur Verfügung.

Wie bei so manch anderer Komponente von SQL Server 2005 gab es die Möglichkeit, Webservices zu implementieren, auch schon früher in Form eines Features Packs: SQLXML genannt. Damit konnte man schon recht komfortabel aus gespeicherten Prozeduren Webdienste machen und diese veröffentlichen. Dabei spielten der Internet Information Server (IIS), ein ISAPI-Filter für den IIS und OLE DB die entscheidenden Rollen. Mit anderen Worten: Der Weg vom Aufruf des Webservice durch einen Client bis hin zur SQL Server Datenbankmaschine war nicht eben kurz und die Leistungsfähigkeit dadurch eingeschränkt. In SQL Server 2005 wird SOAP zu einem First Class Kommunikationskanal von SQL Server. Externe Software ist nicht mehr notwendig um HTTP-Webservices mit SQL Server zu implementieren. SQL Server benötigt einzig den von Windows XP SP2 und Windows 2003 bereitgestellten Kernel-Treiber *http.sys* und daher keinen installierten IIS. Ein Webservice ist mit dem T-SQL-Kommando *CREATE ENDPOINT ... FOR HTTP* schnell angelegt. Gespeicherte Prozeduren und benutzerdefinierte Funktionen können als Methoden eines Webservice vereinbart werden. Außerdem gibt es noch die bequeme – aber gefährliche – Variante, beliebige T-SQL-Batches über den Webservice ausführen zu können. Mit dem Anlegen eines HTTP-Endpunkts ist der Webservice tatsächlich schon fertig. Über den im *CREATE*-Kommando vereinbarten, URI-Pfad kann der Service erreicht werden und bei der Angabe des Parameters *?WSDL* im http-Aufruf liefert der Server brav die Beschreibung des

neuen Service in Form eines Web Services Description Language (WSDL)-XML-Dokuments. Nach dem Konfigurieren der Sicherheit für den Endpunkt ist der Webservices bereit verwendet zu werden. Das ist wirklich einfach!

Service Broker

Der Service Broker ist sicherlich zugleich eine der interessantesten aber auch am wenigsten intuitiv verständlichen Komponenten von SQL Server 2005. Dass Sie die Beschreibung des Service Brokers gerade hier im Abschnitt »Gekoppelte Systeme und SOA« finden, ist (kein) reiner Zufall, denn der Service Broker hat viel mit der Skalierbarkeit von Systemen zu tun, die miteinander kommunizieren. Das können sowohl separate Systeme sein, zum Beispiel zwei Applikationen, die über Webservices miteinander verbunden sind, als auch sehr eng gekoppelte Subkomponenten *eines* Systems. Innerhalb des SQL Servers wird der Service Broker von der Datenbankmaschine selbst verwendet, und es kann unter Umständen sogar Sinn ergeben, in einer Benutzeranwendung den Broker *innerhalb* einer Datenbank einzusetzen. Doch was macht der Service Broker überhaupt?

Die Aufgabe des Service Brokers ist es, asynchrone Kommunikation sicherzustellen. Dabei werden Nachrichten über Warteschlangen gesendet und empfangen. Wenn Sie jetzt sofort an das Microsoft Message Queuing Subsystem von Windows denken (MSMQ), dann liegen Sie gedanklich vollkommen richtig. Konzeptionell geht es hier um das gleiche Thema. Die Bereitstellung dieser Technologie innerhalb SQL Servers hat aber diverse Vorteile:

- Integrierte Einrichtung und Verwaltung: Sie sind als SQL Entwickler nicht darauf angewiesen, zwei verschiedene Systeme einzusetzen. Die Service Broker-Objekte werden in der normalen Routine der Datenbank mit administriert. Die Warteschlangen (die als SQL Server Tabellen implementiert sind) werden mit einem gewöhnlichen Backup gesichert und durch die Hochverfügbarkeitsmechanismen des Servers geschützt.
- Vereinheitlichtes Programmiermodell: T-SQL wird für die Datenbankabfragen *und* die Nachrichtenverarbeitung eingesetzt.
- Transaktionale Nachrichtenverarbeitung: Mit Service Broker Warteschlangen können relativ leicht nachrichtenbasierte verteilte Transaktionen implementiert werden, ohne das ein spezieller Transaktionsmonitor verwendet werden müsste.

Bleibt noch zu klären, wie denn überhaupt Anwendung von der asynchronen Nachrichtenverarbeitung des Service Brokers profitieren können. Es folgen die möglichen Vorteile:

- Verbesserte Antwortzeiten: Prozeduren müssen nicht auf den Abschluss der Bearbeitung nach einem Aufruf eines einen langsamen Teilprozesses warten, sondern können sofort weiter arbeiten. Bei einer Erhöhung der Last werden die Antwortzeiten nicht überproportional länger.
- Lastverteilung: Eine Nachricht in einer Warteschlange kann durch einen Empfänger auf einer anderen Maschine verarbeitet werden.
- Verzögerte Bearbeitung: Nachrichten können in einer Warteschlange gespeichert werden, bis sie von einem Batch Job (Beispielsweise zum Tagesabschluss) verarbeitet werden.
- Parallelisierung: Durch die Verteilung auf mehrere Warteschlangen können nachrichten parallel verarbeitet werden.

- Lose Kopplung: Service Broker Warteschlangen können genau wie Webservices der Kopplung von Anwendungen dienen. In diesem Fall werden natürlich SQL Server basierte Anwendungen verbunden.

Verteilte Daten

Nachdem Sie gerade kennen gelernt haben, wie Sie verteilte Anwendungen verbinden können, wird jetzt die Perspektive ein wenig gedreht und es geht darum, wie SQL Server mit verteilten Daten arbeiten kann.

SQL Server Integration Services

Auf dem Feld des Datenmanagements gibt es Begriff *Extract-Transform-Load (ETL)*, der bestimmte, wiederkehrende Operationen treffend beschreibt: Das Abholen von Daten aus einer Datenquelle, wie einem Host, einer Transferdatei oder einem Datenbankmanagementsystem. Das Transformieren der Daten in ein Format, in welchem diese optimal weiterverwendet werden können. Und das Laden der Daten in ein Zielsystem, wie eine Datenbank, die Teil eines Data Warehouse Systems ist. Die Microsoft Variante von ELT heißt *SQL Server Integration Services (SSIS)* und stellt eine komplette Neuentwicklung für SQL Server 2005 dar.

Daten werden in einer typischen IT-Umgebung häufig in vielen verschiedenen Systemen und in vielen verschiedenen Formaten gespeichert. Möchte man diese externen Daten in eigene Anwendungen importieren, dann ist dieser Vorgang in den allerwenigsten Fällen ein einfaches Kopieren der Daten. Die Daten müssen möglicherweise überhaupt erst einmal abgeholt werden (vielleicht per FTP-Dateitransfer oder durch eine Abfrage eines entfernten Datenbankservers), bereinigt, validiert, von den Datenstrukturen her angepasst und so weiter. Da können eine Menge Programmieraufgaben anfallen und eine Menge Jobs eine Rolle spielen. Die SSIS vereinfachen viele der notwendigen Aufgaben indem Sie eine Vielzahl fertiger Funktionsbausteine »out-of-the-box« mitbringen.

Mit dem SSIS-Designer im Business Intelligence Studio können Sie mächtige ETL-Pakete entwickeln. Die Architekten der SSIS haben sich wirklich Mühe gegeben, clevere und saubere Konzepte zu entwickeln, welche die Arbeit mit den SSIS zu einem Vergnügen machen. Datenfluss- und Kontrollfluss sind sauber voneinander getrennt, und die Datenverbindungen können übersichtlich verwaltet werden. Es steht eine Vielzahl von Transformations-Tasks zur Verfügung, von einfachen Mappings bis hin zu Fuzzy-Logik-Algorithmen. Die Steuerung des Kontrollflusses ist im Grunde eine visuelle Programmieroberfläche, mit Elementen wie Entscheidungsstrukturen, konditionalen Schleifen und Blöcken.

Wir finden die SSIS so gut, dass wir ihnen im Integration Services-Teil dieses Buches vier große Kapitel gewidmet haben.

Replikation

Falls Sie vorhaben, mit SQL Server Filialsysteme zu entwickeln, oder die Daten zwischen zwei Firmenstandorten zu synchronisieren, dann sollten Sie sich unbedingt einmal mit den Replikationsfähigkeiten von SQL Server beschäftigen. Replikationsmechanismen sind eine klassische Methode von Datenbankservern, Informationen, die an verschiedenen Orten liegen, durch Synchronisation auf der

Datenebene abzugleichen. Ziel ist, den Datenzugriff für Anwender und Applikationen möglichst transparent zu halten. Die Programmierer von Datenbankanwendungen sollen sich nach Möglichkeit nicht um die Details der Implementierung der Synchronisation kümmern müssen, sondern so arbeiten können, als hätten Sie es mit einer lokalen Datenbasis zu tun. In SQL Server gibt es verschiedene Arten von Replikation:

- **Transaktionale Replikation:** Dieser Replikationstyp ist besonders gut für verbundene Umgebungen geeignet und kann dann hohe Performance erzielen. Die Latenzzeit zwischen einer Änderung und der Synchronisation ist besonders kurz. Besonders einfach lässt sich diese Art der Replikation in Topologien mit einem zentralen *Verleger* und verteilten *Abonnenten* einrichten, wenn die Daten nur an einer Stelle geändert werden sollen. Prinzipiell sind auch Szenarien denkbar, bei denen an mehreren Orten Änderungen vorgenommen werden – diese Verfahren bieten aber keine Konfliktauflösung an.
- **Merge-Replikation:** Die ist nicht ganz so »flott« wie eine transaktionale Replikation, funktioniert dafür aber auch reibungslos, wenn Abonnenten tagelang offline waren und bietet sowohl vordefinierte und selbst definierbare Konfliktauflösungsverfahren. Damit werden auch sofort die typischen Anwendungsfälle klar: Merge-Replikation lässt sich ausgezeichnet für »Travelling Users« einsetzen. Deswegen ist SQL Server Express Edition auch als Client für die Merge-Replikation einsetzbar und für SQL Server Mobile stellt diese einen wichtigen Weg dar, die Daten zwischen einem Smart Device und einem SQL Server-Verleger abzugleichen.
- **Snapshotreplikation:** Sie ist das Brut-Force-Verfahren unter den Replikationstypen. Wie der Name schon sagt, werden komplette Tabellen oder Teile von Tabellen als Schnappschüsse an die Abonnenten übermittelt. Wegen der großen Datenmengen ist die Übertragung langsam, verteilte Änderungen sind nicht möglich und insgesamt eignet sich das Verfahren vor allem in Spezialsituationen. Wenn immer komplette Batches übertragen werden sollen, wie die Transaktionen eines Tages oder sich selten ändernde Stammdatentabellen und dabei eine möglichst sichere Methode gesucht wird, kommt Snapshotreplikation in Betracht. Eines ist die Snapshotreplikation nämlich ganz sicher: *stabil*.

Bei der Auswahl des richtigen Replikationsverfahrens gibt es einige Parameter zu überdenken. Jedes dieser Verfahren ist in SQL Server unterschiedlich realisiert und dadurch ergeben sich unter anderem technische Einschränkungen oder Auswirkungen auf den Aufbau der Datenbank, die Sie als Entwickler kennen und berücksichtigen sollten. Beispielsweise müssen Sie bei der transaktionalen Replikation jede Tabelle mit einem Primärschlüssel versehen. Das ist natürlich sowieso eine gute Idee, aber vielleicht ist ja gerade Ihre Datenbank anders aufgebaut. Außerdem müssen Sie bei der Programmierung von Triggern ein paar Randbedingungen beachten, die es ohne transaktionale Replikation nicht gäbe. Solche Feinheiten gilt es bei jedem Replikationstyp zu berücksichtigen. Leider hat es das Thema Replikation nicht mehr in dieses Buch geschafft. Das holen wir in einer zukünftigen Ausgabe oder auf der Website nach.

Verteilte Abfragen

Möchten Sie mit Daten aus unterschiedlichen Datenquellen arbeiten, die nicht in Ihrer Serverinstanz, sondern in verschiedenen externen Datenquellen lagern, dann ist es natürlich verlockend, die Daten nicht hin und her schieben zu müssen, sondern sie an Ort und Stelle zu belassen und eine Möglichkeit

zu haben, die Daten *direkt* abzufragen. Diese Möglichkeit gibt es in SQL Server 2005. Es sind die *verteilten Abfragen* oder noch genauer: die verteilten *heterogenen* Abfragen. Verteilte Abfragen basieren auf OLE DB-Datenprovidern. Alles, wofür es einen passenden OLE DB-Provider gibt, kann in Form einer verteilten Abfrage angesprochen werden: von der ASCII-Textdatei, über Access-Datenbanken bis hin zu Datawarehouse-Daten auf einem Teradata Server. Notfalls tut es auch ein ODBC-Treiber, denn für eine ODBC-Datenquelle gibt es einen universellen OLE DB-Provider. Werden die externen Datenquellen regelmäßig abgefragt und sollen die Abfragen sich nahtlos in die lokale T-SQL-Programmierung einfügen, kann man über so genannte *Verbindungsserver* nachdenken. Ein Verbindungsserver ist für die Datenbankmaschine ein logischer Server, hinter dem sich die notwendigen Verbindungsinformationen für verteilte Abfragen verbergen, wie die Parameter für den OLE DB-Provider und die Sicherheitsinformationen. Zusätzlich kann es Ausführungshinweise geben, die den SQL Server bei der Verarbeitung unterstützen. Dabei geht es um Fragen, wie: »Verfügt der entfernte Server über eine kompatible Sortierung der Daten?«

Datenanalyse und Darstellung

Es muss nicht immer eine komplette Datawarehouse-Installation sein, wenn es um die Themen Datenanalyse und Darstellung geht. Fast alle Anwendungen profitieren auf die eine oder andere Art von Berichten und für einige Anwendungen sind *OLAP*-Funktionen sinnvoll. Wobei OLAP für *Online Analytical Processing* steht und damit für die Möglichkeit, auch große Datenmengen schnell und interaktiv zu untersuchen. Von den Business Intelligence Technologien in SQL Server 2005 können viele Anwendungen profitieren. »Offiziell« gehören dazu die schon erwähnten Integration Services, die Reporting Services und die Analysis Services.

Reporting Services

Berichte über die Inhalte einer Datenbank zu erstellen, ist für Entwickler kein großes Kunststück. Schließlich ist in Visual Studio eine Entwicklerversion von Crystal Reports bereits integriert. Wer mehr Funktionalität benötigt, der greift auf die Vollversion oder eines der anderen Reporting-Tools zu. Damit kann man dann in einer Clientanwendung Berichte anzeigen lassen.

Was ist aber, wenn Berichte automatisiert erstellt und verteilt werden sollen? Dies möglichst in verschiedenen Formaten und mit Parametern, die in Abhängigkeit von den Empfängern mit Argumenten versehen werden. Was ist mit dem Erstellen von Versionen von Berichten und dem Caching von Berichtsdaten für lange laufende Abfragen? Hier kommen die serverbasierten Reporting Services ins Spiel. Die sind im Kern als Windows-Dienst und dazu gehörenden SQL Server Datenbanken implementiert. Der Dienst stellt die notwendigen Basisfunktionen zur Verfügung, wie eine Rendering Engine für die verschiedenen Ausgabeformate (HTML, PDF, EXCEL, CSV, TIFF und XML), die Authentifizierungsmechanismen und die Schnittstelle zum Auftragsplanungssystem der Serverinstanz. Anwendungen, wie der Report Designer im Business Intelligence Studio können über Webservices mit dem Report Server kommunizieren, um Berichtsdefinitionen hochzuladen oder einen generierten Bericht abzuholen. Außerdem gibt es eine einfache ASP.NET- basierte Berichtsmanager-Webanwendung, mit der Administratoren die Berichte eines Servers verwalten können oder die als Ausgangspunkt für ein Report-Portal dienen kann.

Berichte können über einen Weblink aufgerufen und in einem passenden Viewer angezeigt werden. Eine simple Variante ist die Verwendung eines Webbrowsers oder eines Webbrowser-Steuerelements. Visual Studio Entwickler haben es mal wieder besser: Die können nämlich einfach das *ReportViewer*-Steuerelement aus der Toolbox einsetzen. In der Standard- und der Enterprise Edition von SQL Servers wird zusätzlich ein vereinfachter Berichtsdesigner angeboten, der es Benutzern ermöglicht, ihre eigenen Berichte zusammenzustellen ohne das sehr technisch ausgerichtete Visual Studio zu benutzen. Grundlagen und die Anwendung der Reporting Services finden Sie in diesem Buch ab Kapitel 35.

Analysis Services

Leistungsfähige und *bezahlbare* Datenbankmanagementsysteme wie SQL Server 2005 sind in Verbindung mit rapide sinkenden Kosten für Speichermedien nicht ganz schuldlos an einem Trend in der IT, der in den letzten Jahren immer mehr spürbar wird: der Datenexplosion in den Unternehmen. Als Microsoft mit SQL Server 7.0 als Showcase die Anwendung Terraserver erfand – eine Bilddatenbank für Satellitenfotos –, da staunte man noch über eine Datenbank, in der sich fast zwei TB Informationen befanden. Auf Fotos aus dieser Zeit (1998) sieht man einen kleinen Schreibtisch als »Konsole«, umgeben von schrankgroßen Racksystemen für die RAID-Arrays. Darin steckten damals 9 GB (!) Festplatten und man brauchte eine Menge davon. Heutzutage (2006) findet man im Consumerbereich bereits Festplatten, von denen eine einzige ein halbes TB fassen kann, um all die vielen MP3-Dateien, DVDs und digitalen Urlaubsvideos zu speichern, die zuhause so anfallen. Natürlich ist die Datendichte bei den Serverplatten ebenfalls ungefähr um den Faktor 30 gestiegen und niemand erschreckt sich mehr.

Im Unternehmensumfeld hat man es weniger mit Streaming Media-Daten zu tun (obwohl einige Administratoren da sicher ganz anderer Meinung sind) als vielmehr mit riesigen Mengen von Informationen über Geschäftsvorfälle. Nachdem inzwischen fast alle Vorgänge elektronifiziert sind, fallen an allen Ecken und Enden Tracking-Informationen an – und die wollen ausgewertet sein. Das sagt jedenfalls mit berechtigtem Interesse die Geschäftsführung.

Die Daten mit den Integration Services einzusammeln und in relationalen Datenbanken zu speichern, ist überhaupt kein Problem. Vielleicht stecken die operationalen Daten ja bereits in einer SQL Server-Datenbank und müssen nur noch ausgewertet werden. Aber da gibt es überraschenderweise Probleme. Mit klassischen relationalen Datenbankservern können Sie hervorragend Transaktionen verarbeiten und Daten sammeln. Die schnelle Auswertung größerer Datenmengen kann dagegen langatmig werden. Eine Datenbank gleichzeitig für eine schnelle Transaktionsverarbeitung und dynamisches und effektives Reporting zu optimieren, ist nahezu unmöglich. Der Aufbau des Datenbankschemas in einem OLTP-System ist auf das reibungslose Einfügen und Bearbeiten von Datenzeilen hin optimiert. Ein normalisiertes Schema ist eine wichtige Voraussetzung für das Funktionieren der Datenbankanwendungen. Auswertungen leiden wegen der großen Anzahl zu verknüpfender Tabellen. Auch die Indizierung kann nur in die eine oder andere Richtung ausgerichtet werden. Während man sich für das schnelle Auffinden und Aggregieren von Daten jede Menge Indizes wünscht, werden die *INSERT*-, *UPDATE*- und *DELETE*-Operationen durch übermäßige Indizierung ausgebremst. Geht es nur um eine begrenzte Anzahl von Werten, die schnell zur Verfügung gestellt werden müssen, dann kann man mit Tricks wie mitlaufenden Spalten arbeiten (gezielte Denormalisierung). Bei einem SQL Server in der Enterprise-Edition helfen indizierte Sichten und partitionierte Tabellen und Indizes weiter.

Dennoch bleibt das Problem bestehen, dass große Berichtsläufe und umfangreiche Auswertungen das gleichzeitige Bearbeiten von Geschäftsvorfällen natürlich beeinträchtigen.

Der (inzwischen) klassische Ansatz für die Lösung dieser Problematik ist die Trennung von Transaktions- und Analysedaten. Tabellen für die Berichterstellung werden in eine zweite Datenbank kopiert. Diese Datenkopien können jetzt sehr dicht indiziert werden und es bietet sich an, die Datenstrukturen für Auswertungen zu optimieren. Im Data Warehousing werden dazu so genannte Stern- und Schneeflocken-Schemata eingesetzt.

Die Analysis Services gehen aber noch einen Schritt weiter und bieten hoch optimierte Datenstrukturen für Analysedaten. Das Datenformat ist nicht relational, sondern *multidimensional*, die Daten werden nicht in Tabellen, sondern in *Würfeln* (engl *Cubes*) gehalten. Auswertungen der Informationen solch eines Würfels gehen in Sekundenschnelle vor sich, und dies selbst bei Datenmengen, bei denen eine normale T-SQL-Abfrage einige Minuten bis Stunden laufen würde. Mit den Analysis Services lässt sich so genanntes *Online Analytical Processing (OLAP)* realisieren. OLAP-Abfragen werden schnell und interaktiv ausgeführt, und es stehen typische Würfel-Operationen, wie *Slicing*, *Pivotierung*, *Drill Down* oder *Roll Up* zur Verfügung. Der Benutzer kann mit Excel oder spezialisierten OLAP-Werkzeugen die Analysen ad hoc durchführen. Das wäre mit »normalen« T-SQL-Befehlen in »normalen« Datenstrukturen nicht so »schwerelos« machbar. Zu jeder neuen Benutzerabfrage würde in diesem Fall ein vermutlich hochkomplexer T-SQL-Befehl generiert und ausgeführt – mit einer Wartezeit von vielen Sekunden oder Minuten. Das kann bei Abfragen, die gegen Analysis Services Cubes laufen, nicht passieren. Die meisten der möglichen Aggregate liegen bereits berechnet im Cube. Die fehlenden Werte können schnell aus Stützwerten berechnet werden.

Wenn Sie Transact-SQL näher betrachten, dann finden Sie im *SELECT*-Befehl Optionen wie *WITH CUBE* oder *WITH ROLLUP*. Tatsächlich sollen damit Würfel-Sichtweisen auf relationale Datenstrukturen ermöglicht werden. Das ist in der Praxis aber untauglich – außer Sie machen sehr gerne und lange Kaffeepausen. *ROLLUP* und *CUBE* berechnen die Aggregate erst zum Zeitpunkt der Abfrageausführung und erzeugen die Rollups und Cubes dann im Arbeitsspeicher (oder der temporären Datenbank des SQL Servers – *tempdb*). Nicht immer eine gute Idee.

Eine gute Idee für das Vorbereiten von Auswertungen ist dagegen der Einsatz der Analysis Services. Im Entwicklerbuch haben diese leider keinen Platz gefunden,³ aber wir möchten die wichtigsten Highlights für Entwickler nicht verschweigen:

- **Speichermodelle:** Die Analysis Services machen verschiedene Angebote für die optimale Datenspeicherung: MOLAP, ROLAP und HOLAP. MOLAP (»M« steht für multidimensional) ist die native Speicherform multidimensionaler Daten. Durch spezielle Indizes und diverse »Speichertricks« ist hier die Anfrageleistung am höchsten. MOLAP-Cubes können aber leider physische Limits erreichen, so dass die anderen Speicherverfahren eingesetzt werden müssen: in ROLAP (»R« = relational) werden die Daten eines Würfels komplett in relationaler Form abgelegt (mit Bergen an Indizes), und HOLAP (»H« = hybrid) stellt einen schönen Kompromiss dar, bei dem nur die vorausberechneten Aggregate in MOLAP gespeichert werden, die Basisdaten aber relational. Beim Entwurf des Speicherlayouts wird mit Unterstützung eines Assistenten ein Kompromiss zwischen

³ Wie legen Ihnen das Buch »Business Intelligence und Reporting mit SQL Server 2005«, Microsoft Press, ISBN 978-3-86063-994-8 als gute Einführung in diese Thematik ans Herz.

Abfrageleistung und Speicherbedarf gefunden werden. *Alle* denkbaren Aggregate und Teilaggregate im Voraus zu berechnen, führt zu einer erneuten Datenexplosion und zwar einer dramatischen. Die »Intelligenz« der Datenspeicherung gehört zur hohen Kunst der Analysis Services.

- **MDX:** SQL ist eine gute Abfragesprache für relationale Daten, kann aber mit Daten in Würfeln rein gar nichts anfangen. Daher wurde die Abfragesprache Multidimensional Expressions (MDX) entwickelt. MDX kennt viele sinnvolle Operatoren und Funktionen, mit denen sich typische Aufgaben wie Slicings, nicht-additive Aggregationen, Zeitreihenvergleiche und vieles mehr durch mächtige Ausdrücke kurz und knapp formulieren lassen. Selbstverständlich kommt MDX mit multidimensionalen Daten gut zurecht und erlaubt eine präzise Definition der Ergebnis-*Würfel*.
- **XML for Analysis (XMLA):** XMLA ist ein SOAP-basiertes Protokoll für den Zugriff auf Würfeldata im Web und anderswo. XMLA ist ein Industriestandard. Derzeit beteiligen sich Microsoft, Hyperion und das SAS-Institut an der Standardisierung.
- **ADOMD.NET:** ADOMD.NET ist der Datenprovider, den Sie für die Kommunikation mit den Analysis Services einsetzen können. ADO.NET kommuniziert über XMLA mit den Services, erlaubt das Senden von MDX-Abfragen an den Server und nimmt Ergebnisse in Form von CellSets, AdomdDataReaders und System.Xml.XmlReaders entgegen. Sie ahnen es sicher: Die ersten beiden Klassen sind die multidimensionalen Geschwister der ADO.NET Datasets und DataReaders.
- **Data Mining Algorithmen:** Berichte, Charts und andere Auswertungen lassen sich relativ leicht erstellen wenn, die OLAP-Würfel erst einmal vorbereitet sind. Dazu muss man aber wissen, welche Maßzahlen (Measures) überhaupt von welchen Faktoren (Dimensionen) abhängig sind. Abhängigkeiten in großen Datenmengen zu finden, ist die Aufgabenstellung des Data Mining. SQL Server bringt je nach Edition bis zu 11 verschiedene Data Mining Algorithmen mit: von Entscheidungsbäumen über Bayes Algorithmen bis hin zu neuronalen Netzwerken.
- **Unified Dimensional Model (UDM):** UDM ist ein Ansatz für die Modellierung, der es ermöglicht, über verschiedene heterogene Datenquellen hinweg ein integriertes Datenmodell zu entwickeln. Für den Benutzer eines UDM-Modells ist transparent, aus welchen Quellen die Informationen stammen, die er analysieren möchte. Potenziell verringert sich dadurch die Notwendigkeit, Daten mehrfach zu halten und zu transportieren (jedenfalls solange es die Auswertungszeiten hergeben).
- **Analysis Services Management Objects (AMO):** Soll die Administration der Analysis Services-Umgebung automatisiert werden, dann können dafür die .NET-Namespaces der Analysis Services Management Objects eingesetzt werden, um Verwaltungsanwendungen zu programmieren.

Natürlich gibt es für einen Analysis Services-Entwickler eine Menge Werkzeuge, Designer und Assistenten für das Aufbauen der Datenstrukturen, das Entwickeln der Objekte, der MDX-Abfragen, das Tuning und so weiter. Die Angebote sind wirklich gut!

Immer Online: 7*24 Betrieb mit SQL Server

Die Datenbank entwerfen und implementieren, die Anwendung programmieren und ausliefern, das ist es – *noch nicht ganz*. Falls Ihre Datenbankanwendung eine unternehmenskritische ist (ist sie doch –

oder?), dann werden Sie sich sicher dafür interessieren, wie man mit SQL Server eine möglichst hohe Verfügbarkeit auf der Datenschicht erreicht. Handelt es sich bei Ihrer Anwendung um eine Mittelschicht- oder Webanwendung (oder beides)? – Dann müssen sich die Überlegungen natürlich auch auf die weiteren Programmschichten erstrecken, die auf den Applikationsservern installiert sind. Bei Windows 2003 Server sollte man sich mit Network Load Balancing (für IP-Anwendungen) und Clustering vertraut machen – aber das ist wieder ein anderes Buch. Für SQL Server gibt es wirklich ein reichhaltiges Angebot von Mechanismen für die Verfügbarkeit und Ausfallsicherheit:

- Database Mirroring
- Server Clustering
- Log Shipping
- Replikation (Merge oder Peer-to-Peer)
- Storage Area Networks (SAN)
- Automatisierte Backup / Restores

Die Verfahren unterscheiden sich unter anderem danach, ob es gemeinsame Ressourcen gibt, die einen möglichen Single-Point-of-Failure darstellen. Das ist zum Beispiel beim Mirroring nicht der Fall, wohl aber beim Clustering. Ein weiteres Unterscheidungskriterium ist die Ebene, auf der die Redundanz gebildet wird. Das ist beim Mirroring die Datenbankebene, beim Clustering eine vollständige Instanz, bei einem SAN sind es die Dateien. Ein automatisiertes Failover bei Nicht-Erreichbarkeit eines Datenbankserver gibt es nur beim Mirroring und beim Clustering. Je nach Budget und Administrationspower sollte es jeder Entwickler in Zusammenarbeit mit den Administratoren schaffen, eine ausfallsichere Lösung auf die Beine zu stellen. Ein heißer Kandidat wird für viele SQL Server 2005-Anwender die Datenbankspiegelung sein. Schauen Sie in Kapitel 20 nach und Sie werden verstehen, warum das so ist.

Desktop und mobile Anwendungen

Nicht immer brauchen Entwickler den großen Datenbankserver für Ihre Anwendungen, der Dutzende von Benutzern parallel bedient. Auch für Anwendungen, die auf einem Desktop laufen, ist es sinnvoll, eine solide Datenbankmaschine einzusetzen anstatt die Daten in eigenen Dateiformaten oder als serialisierte

ADO.NET Datasets zu speichern. Die allermeisten Entwickler arbeiten inzwischen auch genau so, machen sich das Programmiererleben dadurch leichter und verwöhnen ihre Auftraggeber mit exquisiten Zugriffszeiten und gut wartbaren Applikationen. Anders als im Servermarkt der »großen Maschinen«, wo die Zahl der Anbieter sich inzwischen stark konsolidiert hat (auf, na sagen wir, fünf Player) gibt es im Desktop-Bereich noch eine Menge Engines, aus denen man als auswählen könnte. Im Microsoft Umfeld steht darüber hinaus noch die kleine aber feine Office-Software Access zur Verfügung, die es ermöglicht, zügig und mit vergleichsweise geringem Programmieraufwand gute Datenbankanwendungen für Einzelanwender und kleine Organisationen zu erstellen.

SQL Server Express Edition

Für Leute, die mit Visual Studio entwickeln, hat Microsoft schon ab der ersten Version den bevorzugten Weg klar und deutlich aufgezeigt, indem es eine geschrumpfte Version von SQL Server – die SQL Server Desktop Engine (MSDE) – mit der Entwicklungsumgebung gebündelt hat. Mit SQL Server 2005 ist auch die kleine Ausgabe der Serversoftware gründlich renoviert worden und bietet Möglichkeiten, die vormals ausschließlich dem »großen« SQL Server vorbehalten waren. Für kleinere Anwendungen ist die *SQL Server Express Edition* hervorragend geeignet und hat so nebenbei den Vorteil, dass sie keinen Cent Lizenzgebühren kostet. Also gibt es keine Ausrede, Ihre Desktop-Applikation *nicht* auf der Grundlage der Express Edition zu entwickeln und zu vertreiben. In der folgenden Liste finden Sie die wichtigsten Merkmale des »lütten Servers« in knapper Form zusammengestellt.

- **Programmierbarkeit:** Der Transact-SQL Code, den Sie für die Express Edition schreiben, ist zu 100 % auf den größeren Versionen lauffähig. Verwenden Sie .NET-Assemblies, um SQL Server-Objekte zu entwickeln, dann gilt das Gleiche. Sie können Ihre Anwendungen jederzeit völlig schmerzfrei portieren ohne irgendeine Anpassung vornehmen zu müssen. Ziehen Sie ein Backup in der Express Edition und spielen Sie es via Restore ein – das war's. Die Unterschiede betreffen die verfügbaren Features. Den Befehl *CREATE QUEUE* können Sie beispielsweise deswegen nicht ausführen, weil es in der Express Edition eben keinen Service Broker gibt.
- **Performance:** Es wird nur auf einem Prozessor gearbeitet. Das schränkt die Anzahl der gleichzeitig »machbaren« Benutzer und den Durchsatz natürlich ein. Die Express Version darf sich maximal 1 GByte Arbeitsspeicher reservieren und die Größe einer Datenbank ist auf 4 GByte beschränkt. Es gibt keine 64 Bit Version. Sie können aber ein Setup auf einem 64 Bit System vornehmen und der Server wird dann im Windows on Windows-Modus betrieben. Ansonsten gibt es keine künstlichen Bremsen, wie noch zu Zeiten der MSDE. Bei der wurden maximal 5 Clientanfragen gleichzeitig verarbeitet (technisch ausgedrückt waren maximal 5 worker threads parallel aktiv).
- **Keine Auftragsverwaltung:** Einen *Agent* für regelmäßige Wartungstätigkeiten, wie das Sichern der Datenbanken oder das Reindizieren kennt die Express Edition leider nicht. Da sind dann Sie als Entwickler gefragt. Es gibt eine Menge Möglichkeiten, dieses Manko kreativ auszugleichen. Mit T-SQL

oder den Server Management Objects – der .NET-Verwaltungs-API von SQL Server – können Sie jede nur denkbare Aufgabe zur Hege und Pflege einer Datenbank erledigen. Zusätzlich muss nur noch ein Scheduler entwickelt werden, das war's dann auch schon. In Visual Studio gibt es eine Projektvorlage für einen Windows-Dienst. Die bietet sich an.

- **Management Studio Express:** Klar, auf Ihrem Entwicklungsrechner können Sie die SQL Server Express Edition ganz einfach mit dem normalen Management Studio administrieren. Diesen dürfen Sie aber auf gar keinen Fall an Ihre Kunden weitergeben. Um für den Servicefall eine Möglichkeit zu haben, mal eben einen Blick auf die Datenbanken einer Express Edition zu werfen, können Sie Management Studio Express einsetzen. Das gibt es als kostenlosen Download und fühlt sich überall dort zuhause, wo SQL Server Express installiert ist. Sie können es allerdings nicht auf einem Rechner installieren, auf dem bereits das »große« Management Studio läuft.

- **Merge-Replikation:** Gerade für Satellitenanwendungen auf Notebooks, die hin und wieder mit einer zentralen Datenbank synchronisiert werden müssen aber ansonsten völlig getrennt von dieser operieren, bietet sich mit der Merge-Replikation ein SQL Server Mechanismus an, der ein stabiles Framework für den Datenabgleich zur Verfügung stellt. Im Gegensatz zu MSDE gibt es auf SQL Server Express Merge-Replikation nur als Abo (siehe Kapitel 2, Abschnitt »SQL Server Editionen im Vergleich«).
- **XCOPY-Verteilung von Datenbanken:** Wenn Sie neue Anwendungen und Datenbanken an Zielrechnern verteilen möchten, auf denen die Express Edition bereits installiert ist, dann können Sie die Datenbanken einfach durch das Kopieren der primären Datenbankdatei auf die PCs »installieren«. Die Datenbank wird beim ersten Verbinden einer Applikation mit SQL Server Express in den Server eingehängt (*attached*) und die notwendige Protokolldatei generiert, wenn in der Verbindungszeichenfolge das Attribut *AttachDBFilename=Pfad* vorhanden ist. Der Wermutstropfen: Datenbanken, die in einer Replikation eingebunden werden sollen, können *nicht* auf diese Weise verteilt werden.
- **Benutzerinstanzen:** Eine Besonderheit der Express Edition ist die Möglichkeit, mit Benutzerinstanzen zu arbeiten. Diese laufen – ganz konträr zum Prinzip eines Datenbankservers – im Prozessraum der Benutzerapplikation ab und werden mit dem Sicherheitskontext des interaktiven Benutzers versehen. Benutzerinstanzen sind als Methode gedacht, die Datenbankzugriffe in einem deutlich eingeschränkten Sicherheitsumfeld isoliert ablaufen zu lassen. »Bösartige Software« kann so nicht über Methoden von SQL Server in das Betriebssystem vordringen. Eine Benutzerinstanz wird aus einer laufenden Elterninstanz heraus erzeugt, indem in der Verbindungszeichenfolge das Attribut *user instance=true* mit angegeben wird.
- **Volltextsuche:** In der *Advanced Services*-Ausführung wird SQL Server Express nicht nur mit der reinen Datenbankmaschine, sondern zusätzlich mit dem SQL Server Volltextdienst ausgeliefert. Das ist eine prima Angelegenheit, wie ich finde. Volltextindizierung kann man als Entwickler an vielen Stellen gut gebrauchen, bei denen es um das Suchen in großen Textfeldern, XML-Daten oder in Dokumenten, die in einer Datenbank hinterlegt sind, geht.
- **Reporting Services:** Die sind ebenfalls in der *Advanced Services*-Variante zu finden. Sowohl der Report Server, der dem Anzeigen der Berichte dient, wie auch der Berichtsdesigner sind in abgespeckter Version enthalten. Ein phantastisches Feature für Web- und andere Applikationen.

Möchte man die SQL Server Express Edition mit eigenen Anwendungen ausliefern (und dafür ist sie im Grunde ja gedacht), dann ist die Integration in das Setup-Programm auf verschiedene Arten möglich. Sie können die *Setup.exe* mit einem Parameter */Settings* in einer CMD-Datei aufrufen und die Einstellungen in einer INI-Datei übergeben. Sie können ein .NET-Programm schreiben, welches die Klasse *Process* verwendet, um das Setup zu starten, und haben damit alle Möglichkeiten in der Hand, verschiedene Fälle bei der Installation abzudecken (»Gibt es schon ein installiertes SQL Server 2005

auf dem Rechner?«). Die Installation der Express Engine lässt sich sogar in ein Visual Studio Click Once-Deployment-Szenario integrieren.⁴

Die allerneuesten Versionen der SQL Server Express Edition können Sie sich jeweils als Download direkt bei Microsoft besorgen. Aktuelle Service Packs sind bereits integriert. Das vereinfacht die Auslieferung noch einmal. Je nach Aufgabenstellung benötigen Sie ein anderes Paket. Da die Auswahl der Komponenten ein wenig unübersichtlich sein kann, folgt hier eine kurze Übersicht der Downloads:

- **SQL Server 2005 Express Edition:** Das ist die Express-Datenbankmaschine »pur« ohne irgendwelchen zusätzlichen Schnickschnack, wie Verwaltungsoberfläche oder Volltextsuche.
- **SQL Server Management Studio Express:** Das ist die lizenzfreie, abgespeckte Variante der Verwaltungsoberfläche für die Express Edition.
- **SQL Server 2005 Express Edition with Advanced Services:** Das ist das Paket aus der Express Engine, mit den Reporting Services, der Volltextsuche und SQL Server Management Studio Express.
- **Microsoft SQL Server 2005 Express Edition Toolkit SP1:** Sollten Sie keine Entwicklerversion von SQL Server besitzen (die aber nur ein paar Euro kostet), dann können Sie sich über das Toolkit das Business Intelligence Management Studio beschaffen. Außerdem finden Sie im Download ein SDK.
- **SQL Server 2005 Books Online:** Die Dokumentation für die die Express Edition ist inzwischen in die Gesamtdokumentation integriert. Bei Bedarf gibt es deren aktuelle Version ebenfalls als Download.

Nach dem Herunterladen sollten Sie sich Online bei Microsoft registrieren. Damit sind Sie rechtlich im grünen Bereich und Sie können sofort damit beginnen, Ihre Anwendungen mit der Express Edition zu vertreiben.

SQL Server Mobile Edition

Was Desktop Applikationen recht ist, kann Anwendungen für mobile Geräte nur billig sein: die Verwendung von »richtigen« Datenbanken als Datenspeicher. Die Argumente sind genau diejenigen, die für den Einsatz der Express Edition im Desktop-Umfeld sprechen: leichteres Datenmanagement und mehr Komfort beim Programmieren. Einen *vollständigen* Datenbankserver wird man kaum brauchen – eine Verwaltung verschiedener Benutzer und ein Rechtesystem ist auf einem Smartphone doch recht überflüssig. Andere Funktionen eines Datenbankservers sind dagegen hoch willkommen: hocheffiziente Datenspeicherung, SQL als Abfragesprache, ja sogar Transaktionen können Sinn machen.

SQL Server Mobile Edition bietet eine Menge an Funktionalität. Das ist erstaunlich – vor allem wenn man in Betracht zieht, dass sie mit einem wirklich kleinen Memory-Footprint (ca. 2–3 MB) auskommt. Speicherplatz ist auf Smart Devices im Moment noch die Mangelressource Nummer eins. Die Mobile Edition läuft unter verschiedenen Betriebssystemen:

⁴ In den Materialien auf der Begleit-CD-ROM finden Sie ein gutes Whitepaper, welches das Vorgehen beschreibt.

- Windows CE 5.0
- Windows Mobile Version 5.0
- Windows Mobile Version 5.0 Smart Phone Edition
- Windows Mobile 2003 (SE)
- Windows XP Tablet PC Edition

Um Daten zwischen dem mobilen SQL Server und einem stationären Gerät austauschen zu können, wird ActiveSync (ab Version 4.0) oder die Websynchronisierung benötigt. SQL Server Management Studio ist prachtvoll dazu geeignet, mobile SQL Server-Datenbanken zu verwalten und mit Visual Studio 2005 ist die Entwicklung im Grund ein Kinderspiel. In Kapitel 28 finden Sie eine Übersicht über die wesentlichen Merkmale der Mobile Edition, damit Sie deren Leistungsfähigkeit besser beurteilen können. Und natürlich wird vorgestellt, wie Sie eine mobile Anwendung entwickeln.