

Intelligenter Schubkasten

Was ändert sich mit dem MS SQL Server 2005?

Im Gegensatz zur Wahl der Entwicklungsumgebung – mit der in der Regel nur der Entwickler zu tun bekommt – spielen bei der Wahl einer SQL-Datenbank gleich mehrere Faktoren eine entscheidende Rolle. Deswegen lohnt sich ein Blick darauf, welche Überraschungen die neue Version des MS SQL Server aus verschiedenen Betrachtungswinkeln bereithält.

von Andreas Kosch

Aus der Sichtweise des Kunden – der die Software-Entwicklung letztendlich bezahlen muss – spielt die Wahl der Entwicklungsumgebung nur eine sehr untergeordnete Rolle. Sobald die (hoffentlich fehlerfreie) Anwendung auf seinem Webserver liegt, ist es ihm egal, in welcher Sprache das Ganze entwickelt und mit welchem Tool die ausführbare Anwendung kompiliert wurde. Völlig anders sieht es jedoch immer dann aus, wenn die Anwendung neben den *.aspx*-Dateien auch aus einer SQL-Datenbank besteht. Denn in diesem Fall bekommt es der Kunde auch im Alltag mit dem verwendeten SQL Server zu tun, so dass die Frage „Welcher SQL Server darf es denn sein?“ wohl überlegt sein muss. Erschwerend kommt in diesem Kontext hinzu, dass viele Kunden einen bestimmten SQL Server bevorzugen (der legendäre „Hausstandard“), sodass der Köcher mit eigenen Argumenten für den MS SQL Server 2005 gut gefüllt sein muss. Denn nur dann, wenn die eigene Anwendung

den speziellen Fähigkeiten eines bestimmten SQL Server auf den Leib geschrieben werden kann, unterliegt man nicht den Einschränkungen des Prinzips des kleinsten gemeinsamen Nenners. Schauen wir uns daher an, warum die Entscheidung für den SQL Server 2005, aus allen Blickwinkeln betrachtet, die richtige Wahl ist. Als fiktives Szenario muss dazu ein kontinuierlich wachsendes und mittlerweile auf verschiedene Standorte verstreutes mittelständisches Unternehmen erhalten, das seit kurzem auch das Internet-Feld (B2B) beackern muss. Der Firmenchef will nun eine neue Software haben, die den aktuellen Herausforderungen gewachsen ist und die bei Bedarf flexibel (also kurzfristig) erweitert werden kann. Selbstverständlich soll das Ganze dann auch nicht die Welt kosten.

Argumente für den Architekten

Die erste Person, die in meinem fiktiven Rollenspiel die Qual der Wahl hat, ist der Architekt der IT-Anwendung. Er muss die sich aus dem Geschäftsprozess des Kunden ergebenden Anforderungen so unter einen Hut bringen, dass alle Anforderungen innerhalb des vom Kunden vorgegebenen Zeit- und Kostenrahmens mit der konzipierten Anwendungsarchitektur umgesetzt werden können. Aus seiner Sichtweise ist es wichtig, dass bereits der verwendete SQL Server einen möglichst großen Teil der gestellten Anforderungen ab-

decken kann, denn dann verringert sich der eigene Entwicklungsaufwand. Aus diesem Blickwinkel betrachtet sprechen zahlreiche Punkte für den MS SQL Server 2005:

- Die Integration der CLR in die Datenbank-Objekte (Stored Procedure, Trigger, UDF) erlaubt sowohl den direkten Rückgriff auf den Funktionsumfang des .NET Framework als auch das eigene Nachrüsten von Funktionen direkt in der Datenbankschicht.
- Der neue Transaction Isolation Level SNAPSHOT erlaubt eine völlig neue Sichtweise auf die von mehreren Anwendern gleichzeitig genutzten Daten. Es ist nun nicht mehr unbedingt notwendig, für lang andauernde Abfragen einen zweiten SQL Server vorzusehen, auf dem die Daten des Vortags eingespielt werden. Über den neuen Abschottungsmodus SNAPSHOT der Transaktion können die Daten auch dann ermittelt werden, wenn gleichzeitig andere Anwender diese aktualisieren.
- Der SQL Server Broker stellt eine skalierbare Plattform für das Austauschen von asynchronen Botschaften mit anderen (internen und externen) Prozessen zur Verfügung. Diese Botschaften können sowohl der eigenen Datenbank, einer anderen Datenbank auf dem gleichen SQL Server oder gar einem externen SQL Server zugestellt werden.

kurz & bündig

Inhalt

MS SQL Server 2005

Zusammenfassung

Die Gründe, warum der MS SQL Server 2005 in den eigenen Projekten sinnvoll eingesetzt werden kann



Abb. 1: Hilfe in der Not – die Dokumentation

- Die Reporting Services stellen ein sofort einsatzbereites Fundament für das Erstellen, Verteilen und Verwalten von Berichten zur Verfügung. Der Nutzer dieser Berichte kann diese wahlweise im *Push*- oder *Pull*-Modus in verschiedenen Ausgabeformaten abrufen, wobei neben dem Browser-Zugriff auch die Integration in die eigene Anwendung unterstützt wird.
- Die Notification Services stellen einen Benachrichtigungsdienst zur Verfügung, der über das Publisher-Subscriber-Verfahren über bestimmte Ereignisse informiert.
- Die direkte Unterstützung von Web Services stellt zusätzlich zum IIS eine weitere Option für den Datenaustausch dar.
- Der native XML-Datentyp ist aus Sicht der geforderten B2B-Anbindung interessant, da nun ein XML-Dokument direkt in der Tabelle gespeichert und ausgewertet werden kann.
- In Bezug auf die Leistung muss sich der MS SQL Server nicht hinter dem Wettbewerb verstecken, er hat seine Internet-Tauglichkeit bereits in Gestalt der 2000er-Version seit Jahren in zahllosen ASP- und ASP.NET-Anwendungen unter Beweis gestellt.

Eine zu entwickelnde Anwendung, die alle diese Fähigkeiten des SQL Server 2005 konsequent ausnutzt, wird schneller fertig werden. Denn zum einen verringert sich der eigene Entwicklungsaufwand und zum

anderen ist auch der Testaufwand und nicht zu vergessen die Bug-Wahrscheinlichkeit wesentlich geringer.

Denn noch immer gilt der Grundsatz, dass der beste Weg zur Vermeidung von Bugs darin besteht, so wenige Programmzeilen wie nur möglich selbst zu schreiben.

Sichtweise des Entwicklers

Für den Entwickler der Anwendung – der die Ideen des Architekten in das konkrete Produkt umsetzen muss – spielt die Produktivität eine entscheidende Rolle. Und an dieser Stelle fällt auf, dass als Alleinstellungsmerkmal, die Integration des MS SQL Server in Visual Studio, sehr weit ausgeprägt ist. Dies bringt im Entwickleralltag sehr viele Vorteile (Zeitersparnis) mit sich, wodurch sich der Vorsprung gegenüber anderen SQL-Datenbanken weiter ausbaut. Da diese Integration auch in den bisherigen Visual-Studio-.NET-Versionen vorhanden war, hier nur eine kurze Übersicht:

- In Gestalt des SQL Server .NET Data Provider stellt das .NET Framework eine Sammlung von Klassen zur Verfügung, deren Fähigkeiten explizit für den MS SQL Server optimiert wurden. Zusätzlich stellt der Namespace *System.Data.SqlTypes* spezielle Strukturen und Klassen für die nativen SQL-Server-Datentypen bereit.

- Der Wizard des *SqlDataAdapter* ist nicht nur in der Lage, bereits vorhandene gespeicherte Prozeduren einzubinden, er kann sogar für das Lesen, Einfügen, Ändern oder Löschen von Datensätzen jeweils eine neue Stored Procedure anlegen, wobei im Programm gleichzeitig die Parameter-Kollektionen der darunterliegenden *SqlCommand*-Instanzen fix und fertig konfiguriert werden.
- Wenn der Solution von Visual Studio .NET ein Datenbankprojekt hinzugefügt wird, kann man direkt aus dem Server Explorer heraus die SQL-Skripte für das Anlegen der Datenbank erzeugen lassen. Dabei werden sowohl alle eigenen Tabellen, die Primär- und Fremdschlüssel, die Indizes als auch die erweiterten Eigenschaften berücksichtigt. Falls auch die Datensätze der Tabellen benötigt werden, steht im Server Explorer von VS .NET 2003 die Export-Data-Funktion zur Verfügung, um die Datensätze in eine Datei zu entladen. Da im Solution Explorer auch eine Kommandodatei generiert werden kann, die aus all diesen einzelnen Dateien eine neue (vorbelegte) Datenbank erstellt, verliert diese Weitergabe der Datenbank im Setup-Projekt ihren Schrecken.
- Über die SQL-Server-2005-Project-Vorlage stellt Visual Studio 2005 einen bequemen Ausgangspunkt für das Entwickeln von Datenbankobjekten zur Verfügung.

Im speziellen Fall einer ASP.NET-2.0-Anwendung sieht es zwar auf den ersten Blick so aus, als ob mit dem *SqlDataSource*-Zugriffsweg verschiedene SQL-Datenbanken gleichermaßen unterstützt werden, aber spätestens bei der *SqlCacheDependencies*-Klasse wird wieder deutlich, wie tief die Integration geht und welche Vorteile daraus für die Anwendung entstehen. Während bei anderen SQL-Datenbanken oder auch älteren MS-SQL-Server-Versionen neben einem Trigger auch eine Hilfstabelle benötigt wird, die regelmäßig von der eigenen ASP.NET-Anwendung im Poll-Verfahren abzufragen ist, kommt ASP .NET 2.0 immer dann ohne diesen Aufwand auf, wenn die Cache-Abhängigkeit auf einer Tabelle (Datensatz) des SQL Server 2005 beruht. Im besten Fall reduziert sich der ganze Aufwand auf nur eine einzige Programmzeile:

```
Cache.Insert("ProductsDataSet", ds,
new SqlCacheDependency(
"Northwind", "Products")
);
```

Entwickler können nur dann produktiv arbeiten, wenn auch die Dokumentation entsprechend ausgebaut ist. Und an dieser Stelle fällt auf, dass die Hilfedateien der neuen Version ausführlicher zusammengestellt wurden, als das beim Vorgänger der Fall war (bei dem es ja mitten drin ein separates Update der Hilfedateien gab). Außerdem enthält die CD der MSDN Library zahlreiche Artikel zum Thema MS SQL Server, sodass die Dokumentation ebenfalls ein starker Pluspunkt ist.

Auch im .NET-Zeitalter behält die Grundregel „So viel wie möglich direkt in der Datenbank implementieren!“ für skalierbare Anwendungen ihre Gültigkeit. Die Erweiterung des T-SQL-Sprachumfangs (Transact-SQL) im SQL Server 2005 macht deutlich, dass die sich aus der .NET-Anbindung ergebenden neuen Möglichkeiten nur das Sahnehäubchen sind, aber diese niemals die Bedeutung von T-SQL in den Hintergrund drängen werden. Auch im SQL Server 2005 ist es sinnvoll, so viel wie nur möglich direkt in T-SQL zu erledigen. Der Rückgriff auf die CLR durch das Einbinden von Methoden aus einer .NET-Assembly ist nur dort sinnvoll, wo eine Aufgabe erledigt werden muss, die sich auf dem bisherigen Weg (T-SQL) entweder gar nicht oder nur mit einigen Klümpchen umsetzen lässt. Widerstehen Sie also der Versuchung, alles Bewährte durch .NET-Aufrufe zu ersetzen. Die ersten Tests, bei denen neben dem Implementierungsaufwand auch die Performanz verglichen wird, werden schnell für Ernüchterung sorgen. Im Bezug auf die Geschwindigkeit wird T-SQL auch mittelfristig unerreicht sein, auch wenn die aktuelle Beta-Version des SQL Server 2005 vermutlich noch nicht im höchsten Gang betrieben wird. Wann ergibt dann der Rückgriff auf die Fähigkeiten der .NET-Framework-Klassen einen Sinn? Zumindest immer dort, wo bisher die so genannten Extended Stored Procedures genutzt wurden oder wo der Zugriff auf externe Ressourcen benötigt wird, um nur zwei Beispiele zu nennen.

Bei den Neuheiten von T-SQL gibt es einige Sachen, die wohl die meisten Ent-

wickler nur selten benötigen werden. Allerdings tauchen auch solche Verbesserungen auf, die sehnsüchtig erwartet wurden, weil diese in nahezu jeder Datenbank von Nutzen sind. Schauen Sie sich dazu einmal die folgende gespeicherte Prozedur an:

```
CREATE PROCEDURE spTestTOP
(
    @iCount INTEGER
)
AS
SELECT TOP(@iCount) *
FROM dbo.TestTbl
ORDER BY recid
RETURN @@ROWCOUNT
GO
```

Die *SELECT-TOP*-Anweisung für das Begrenzen der Ergebnismenge einer *SELECT*-Abfrage wurde so aufgebohrt, dass

T-SQL bietet einen Try-Catch-Weg an

nun auch die Anzahl der maximal zurückzuliefernden Datensätze als Parameter übergeben werden kann. Doch damit nicht genug, ab dem Jahrgang 2005 unterstützt sogar die *UPDATE*- oder *DELETE*-Anweisung die *TOP*-Einschränkung.

Bevor Sie nun gleich abwinken, sei mir der Hinweis gestattet, dass diese Begrenzung in umfangreichen Datenmengen durchaus sinnvoll ist, um das Eskalieren der Sperrungen auf Tabellenebene sowie das extensive Auffüllen der Log-Datei zu vermeiden. Das nun folgende Beispiel löscht die Datensätze in kleinen Häppchen zu je zehn Datensätzen aus der Tabelle, die Endlosschleife wird dabei sofort dann verlassen, wenn beim letzten Mal weniger als zehn Datensätze betroffen waren:

```
WHILE 1=1
BEGIN
DELETE TOP(10)
FROM dbo.DeleteTest
WHERE recid < 100
IF @@rowcount < 10 BREAK
PRINT '10 Datensätze gelöscht'
END
```

Konnten Sie sich bisher mit den strengen Regeln bei der Fehlerauswertung über *@@ERROR* anfreunden? An dieser Stelle mussten Entwickler die Dokumentation ganz penibel lesen, um sich nicht aus Versehen ins eigene Knie zu schießen. Gerade bei der Frage, wann die Fehlerkennzeichnung über *@@ERROR* wieder automatisch zurückgesetzt wird und in welchem Gültigkeitsbereich des Aufrufs dieser Wert überhaupt auslesbar war, machten Einsteiger immer wieder die gleichen Fehler (allerdings sind auch erfahrene Entwickler davor nicht gefeit) – mit dem neuen Server ist auch an dieser Stelle eine gravierende Besserung in Sicht. In T-SQL steht nun auch der gleiche *TRY-CATCH*-Weg zur Verfügung, der seinen Siegeszug in den meisten Programmiersprachen angetreten hat. Das folgende Beispiel versucht in einem *TRY*-Abschnitt, bei einem *INSERT*-Aufruf den Zustand *NULL* in eine Spalte zu schreiben, die bei der *CREATE-TABLE*-Anweisung als *NOT NULL* gekennzeichnet wurde. Über die Funktion *ERROR_NUMBER* kann das T-SQL-Skript im *CATCH*-Abschnitt die exakte Fehlernummer des SQL Server auslesen, um dann die Transaktion kontrolliert zurückzusetzen (Listing 1).

Mit diesen exemplarisch herausgezogenen Beispielen endet der Exkurs in die T-SQL-Neuheiten. In [1] finden Sie eine vollständige Zusammenstellung der Erweiterungen, wobei dort z.B. auch über mehrere PDF-Dokumente ausführlich erläutert wird, wozu man den neuen *SNAPSHOT*-Isolationsgrad für die Datenbanktransaktion in der Praxis benötigt.

Listing 1

```
BEGIN TRY
BEGIN TRANSACTION
INSERT INTO dbo.TestTbl (wert) VALUES (NULL)
PRINT 'Geht das gut?'
COMMIT TRANSACTION
END TRY

BEGIN CATCH
DECLARE @Error AS INTEGER
SET @Error = ERROR_NUMBER()
ROLLBACK TRANSACTION
IF @Error = 515 PRINT 'NULL geht nicht'
ELSE PRINT ERROR_MESSAGE()
END CATCH
```

Datenbank-Administrator

Für den vom Kunden angestellten Datenbank-Administrator der Firma ist vor allem wichtig, dass die Anwendung störungsfrei und ohne Ausfall läuft. Da man jedoch immer mit Hardwareausfällen rechnen muss, stellt eine Internetanwendung, die von auf verschiedene Zeitzonen verteilten Anwendern genutzt wird, eine besondere Herausforderung dar. In Verbindung mit Windows Server 2003 unterstützt der SQL Server 2005 das so genannte *Failover Clustering*, bei dem im Störfall ein automatischer Wechsel auf eine andere Maschine erfolgt.

Wenn man nur die Verfügbarkeit der Datenbank betrachtet, gibt es mit dem *Database Mirroring* eine Alternative. Wird diese Option aktiviert, wird jeder Schreibzugriff einer Transaktion (die Änderungen der Log-Datei) auf einem zugewiesenen Reserve Server gespiegelt. Wenn in diesem Szenario der primäre SQL Server ausfällt, kann die Anwendung sofort die Datenbankverbindung auf den Reserve Server umleiten. Der Vorteil gegenüber einem

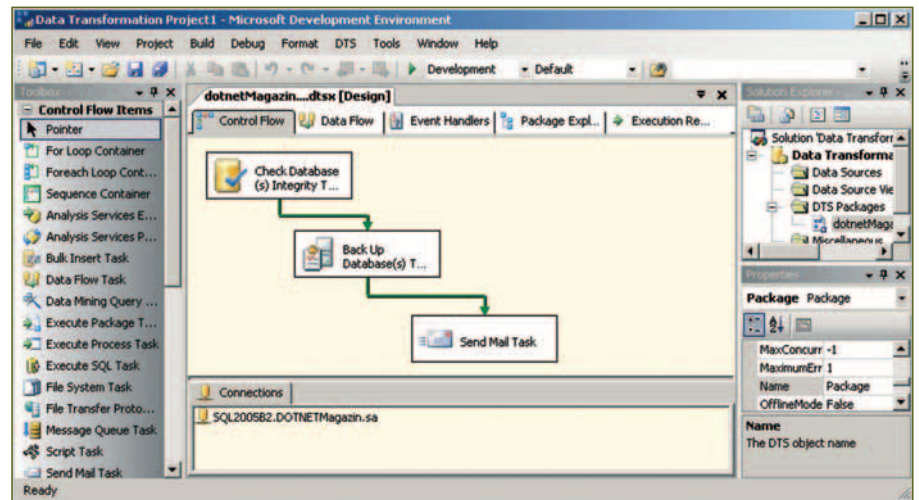


Abb. 2: Business Intelligence Development Studio

Cluster-System ist, dass beide Server ständig online sind und synchronisiert werden, sodass man zum Lastausgleich auch im Normalbetrieb bestimmte Abfragen vom Reserve Server abarbeiten lassen kann.

Neben den unplanbaren Störungen gibt es jedoch auch geplante Wartungsarbeiten. Und hier wurde der SQL Server

2005 an verschiedenen Stellen so erweitert, dass nach einer Änderung kein Neustart mehr erforderlich ist und der Server somit während der Zeitdauer der Wartung ständig online bleibt. Das fängt bei der Speicherzuordnung an, führt über den Neuaufbau von Indizes bis hin zum Wiederherstellen einer Datenbank.

Jetzt abonnieren und Vorteile sichern!

Riesen-Poster!



inklusive Jahres-CD



128 MB USB-Stick!

Ihre Abo-Vorteile:

- Sie erhalten den 128 MB dot.net-USB-Stick*
- Sie sparen rund 10% gegenüber dem Einzelverkauf
- Sie erhalten das **Riesen-Poster** „.NET Framework 2.0“
- Mit der **Jahres-CD** erhalten Sie alle Ausgaben des vergangenen Jahres noch gratis obendrauf
- Sie verpassen keine Ausgabe
- Jede Ausgabe inklusive **Heft-CD** mit vielen Tools

* Nur solange der Vorrat reicht! Memory Stick wird nach Ausgleich der Abonnementrechnung zugesendet.

Der neue SQL Server bringt auch eine neue Werkzeugkiste mit. Alle diejenigen, die sich zu sehr an den Enterprise Manager gewöhnt haben, müssen nun beim neuen SQL Server Management Studio etwas umlernen. Die zentrale Schaltstelle integriert die Funktionen, die zur Verwaltung sowie zur Entwicklung benötigt werden. Ähnlichkeiten mit Visual Studio 2005 sind nicht (!) zufällig. So zeigt zum Beispiel der Editor für SQL-Anweisungen (der eingebettete Ersatz für den Query Analyzer des 2000er-Server) alle ungespeicherten Änderungen über einen gelben Anstrich am linken Fensterrand an. Das gleiche Verhalten hat auch der Editor von Visual Studio 2005.

Ein Highlight des „alten“ SQL Server 2000 war der DTS-Import/Export-Assistent. Trotzdem hat Microsoft sogar an dieser Stelle zwei gravierende Veränderungen vorgenommen. Zum einen kann der Experte nun nicht mehr Objekte zwischen verschiedenen Datenbanken kopieren und zum anderen ist auch die Transformationsfähigkeit über Scripte auf der Strecke geblieben. Das hört sich auf den ersten Blick nach einer Verschlimmbesserung an, aber hinter den Kulissen hat Microsoft nur etwas aufgeräumt. Denn bisher standen beim 2000er-Server gleich zwei Funktionen zum Kopieren von Datenbankobjekten zur Verfügung. Neben dem DTS war auch der Copy Database Wizard für diese Aufgabe zuständig. Der neue Server schafft somit mehr Übersichtlichkeit, indem die drei Aufgaben auf verschiedene Tools aufgeteilt werden. Der neue DTS-Import/Export-Assistent kümmert sich nur noch um die Aufgaben, die seinem Namen entsprechen. Im Gegensatz dazu ist für das Kopieren von vollständigen Datenbanken sowie von Teilen von Datenbanken der Copy Database Wizard zuständig.

Und wenn es darum geht, beim Kopieren auch komplexe Transformationen ablaufen zu lassen, ist nun der in das Business Intelligence Development Studio (Visual Studio) eingebettete DTS Designer (siehe Abbildung 2) das richtige Werkzeug. Dieser ist auf die ETL-Anforderungen (Extraction, Transformation and Loading) von real vorkommenden Datenbanken besser ausgerichtet, als das bei der 2000er-Version der Fall war.

Wenn Sie zum Beispiel über den DTS-Import/Export-Assistenten die Datensät-

ze aus einer CSV-Datei (*comma-separated value*) importieren, stoßen Sie bereits auf die nächste Neuheit. Anstelle des OLE DB Provider greift die neue Fassung auf den Adapter Flat File Source zurück, der das Hantieren mit Datumsformaten und Zeichensätzen vereinfacht. Ist das Paket fertig, kann es nun auch im XML-Format

XML-Format wird unterstützt

gespeichert werden. Die alte Fassung hat dazu noch auf eine binäre OLE-Structured-Storage-Datei zurückgegriffen.

Eine weitere lang ersehnte Verbesserung besteht darin, dass beim sofortigen Ausführen des Pakets viel detaillierte Status- und Fehlermeldungen auf Wunsch abrufbar sind und im Fall eines Fehlers die Korrektur sofort durch das Zurückblättern vorgenommen werden kann, ohne im Wizard alle Einstellungen von Anfang an wiederholen zu müssen.

Was kostet der Spaß?

Es kann nun der Eindruck entstanden sein, dass der SQL Server 2005 für das eigene Projekt zu „overdressed“ ist. Nicht jeder wird für seine ASP.NET-Anwendung gleich einen eigenen Cluster benötigen. Aber auch derartige Bedenken sind kein Argument gegen den SQL Server 2005. Denn es ist immer sinnvoll, sich Spielraum nach oben frei zu halten und zukünftig steht ja mit dem SQL Server 2005 Express auch ein Ableger zur Verfügung, der ideal auf den „kleinen Bedarf“ zurechtgeschnitten ist. Der Steckbrief vom SQL Server 2005 Express wird viele überzeugen:

- Kostenloser SQL Server, der auch kostenfrei weitergegeben werden darf.
- Patches werden vom Windows-Update-Dienst mit eingearbeitet.
- Keine Leistungsbremse bei mehr als fünf gleichzeitigen Benutzern.
- Unterstützt Datenbanken bis zur Größe von 4 GByte - die Log-Dateien und die Datenbank *tempdb* zählen dabei nicht mit.
- Nutzt immer nur eine CPU im Rechner.
- Nutzt nur das 1 GByte vom vorgefundenen Arbeitsspeicher.

- Nutzt die CLR, sodass sich der Funktionsumfang sowie die Programmierbarkeit drastisch erhöht.
- Unterstützt in der Voreinstellung das automatische Ein- und Aushängen von Datenbanken (Funktion *AttachDBFilename* und Datenbankattribut *AUTO_CLOSE*), sodass auch für die Datenbankdatei selbst das XCOPY-Installationsprinzip gilt.

Gerade der letzte Punkt führt dazu, dass das Hantieren mit dem MS SQL Server 2005 Express in der Praxis genauso einfach ist, wie das bisher bei einer ACCESS-Datenbank (*.MDB) war. Es gibt selbstverständlich einen Haken bei der Sache, der erstmalige Zugriff auf die Datenbank muss diese erst einhängen. Wenn sich die Wartezeit also zu sehr störend erweisen sollte, kann allerdings die Voreinstellung für diese Datenbank geändert werden.

Das Fazit besteht nur aus einem einzigen Satz: Es lohnt sich auf jedem Fall, die hoffentlich noch in diesem Jahr erscheinende Endversion auch in den eigenen Anwendungen zu nutzen. Bis dahin lindern die Beta-Versionen des MS SQL Server 2005 und Visual Studio 2005 die sich schnell einstellenden Entzugserscheinungen. Wenn man dann noch berücksichtigt, dass mit der angekündigten Visual Web Developer 2005 Express Edition eine sehr preiswerte Entwicklungsumgebung für ASP.NET-Anwendungen verfügbar sein wird, muss man kein Prophet sein, um die Kombination von Windows Server 2003 Web Edition, Visual Web Developer 2005 Express Edition und MS SQL Server 2005 Express als Nachbrenner für die Verbreitung von ASP.NET vorherzusagen. Ob auch andere Web-Technologien auf den neuen SQL Server setzen, bleibt jedoch abzuwarten. ●

Andreas Kosch (Baujahr 1962) beschäftigt sich in seinem Alltag mit der Konzeption und Entwicklung von dreischichtigen Datenbank Anwendungen für MS SQL Server (Schwerpunkt: .NET Enterprise Services). Als alter Hase hat er es mit mehreren Sprachen (SQL Windows, Delphi, C#, VB.NET) zu tun. Sie erreichen ihn per E-Mail unter: OssiSoft@aol.com.

Links & Literatur

- [1] www.microsoft.com/sql/2005/