SQL DMO – die richtige Code-Library

Sammeln von Testdaten leicht gemacht, Teil 2

von Urs Gehrig

Haben Sie mit Ihrer C#-Applikation schon einmal nach allen auf Ihrem Netzwerk installierten SQL Servern gesucht oder mussten Sie diverse Metadaten Ihres SQL Server mit Ihrer Programmlogik verarbeiten? Wenn Sie da nicht SQL DMO kennen, haben Sie sich dabei sicherlich die Zähne ausgebissen oder zumindest graue Haare bekommen. Im ersten Teil dieser Serie lernten Sie Clone-DB kennen – eine Applikation zum Skripten von Testfällen aus einer produktiven Datenbank. In diesem Teil lernen Sie, wie Sie CloneDB mithilfe von SQL DMO rasch selber schreiben.

Das Zauberwort aus dem ersten Teil dieser Serie lautete *SQL DMO*. Die Menge an Zuschriften, die der Autor dieses Beitrages nach dem ersten Teil erhalten hat, zeigt es deutlich: Die Probleme von *CloneDB* sind weit bekannt, aber kaum jemand kennt *SQL DMO*. Dabei wäre alles gar nicht so schwer, wenn denn nur die richtige Code-Library zur Hand wäre. Wie sich *CloneDB* als C#-Applikation mit einer *SQL-DMO*-Integration dem Benutzer präsentiert, ist in Abbildung 1 ersichtlich. Auf der linken Seite ist der *Server-Objektbaum* – hier finden Sie alle im Netzwerk auffindbaren SQL Server, inklusive deren Datenbanken, Tabellen, *Views*, *Stored Procedures* und *User Defined Functions*. Die rechte Seite von *CloneDB* ist den verschiedenen Ansichten der Serverobjekte vorbehalten – z.B. dem *Insert*-Skript der Tabellendaten, dem *Create*-Skript für ein Datenbankobjekt, den Abhängigkeiten eines Datenbankobjektes oder den Daten einer Tabelle selber. Nützlich für das Debuggen ist auch die *About Box* – hier sehen Sie alle bis dahin geladenen Assemblies mit ihrer Version. (Den Quellcode für *CloneDB* finden Sie auf der Heft-CD.)

kurz & bündig

Inhalt

Im ersten Teil dieser Serie haben Sie die Anforderungen an *CloneDB* kennen gelernt – jetzt werden Sie sehen, wie Sie *CloneDB* mithilfe von *SQL DMO* in C# programmieren

Zusammenfassung

SQL DMO ist eine effizientes API für den SQL Server

Quellcode

C#

Quellcode auf **CD**

Erste Schritte mit SQL DMO

Bevor Sie *SQL DMO* in Ihrem C#-Programm verwenden, müssen Sie Ihrem Projekt eine Referenz hinzufügen (Abbildung 2) – wählen Sie hierzu den Tab COM. Standardmäßig befindet sich die Library *sqldmo.dll* im Verzeichnis *\Programme\Microsoft SQL Server\80\Tools\Binn*. Fügen Sie in Ihrer *using Region* folgende Zeile hinzu:

```
using SQLDMO;
```

Jetzt können Sie mit der eigentlichen Arbeit beginnen: Versuchen Sie sich als Erstes am Auflisten aller verfügbaren SQL

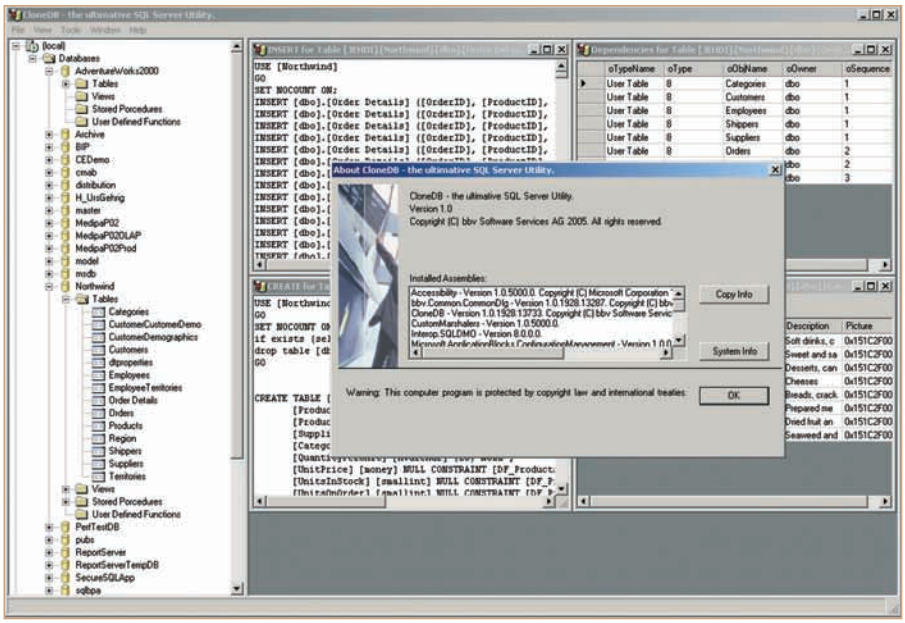

Abb. 1: So präsentiert sich *CloneDB*

Server (Listing 1 auf Heft-CD). Hierfür hält das Objekt *SQLDMO.Application* eine Methode *ListAvailableSQLServers* bereit. Als Rückgabewert erhalten Sie eine *SQLDMO.NameList*, durch welche Sie mit einer *foreach*-Schleife iterieren. Für jeden auf dem Netzwerk gefundenen SQL Server erhalten Sie dessen Host-Namen. Mit diesem alleine können Sie aber noch nicht viel anfangen – wenn Sie mit dem Server arbeiten möchten, brauchen Sie ein Serverobjekt:

```
SQLServer2 myServer = new SQLServer2();
```

Wenn Sie in diesem Serverobjekt die Attribute *LoginSecure* und *HostName* setzen, sind Sie schon fast fertig. Setzen Sie noch das Attribut *ApplicationName* und Sie filtern im SQL Profiler leicht die von *CloneDB* herrührenden Events heraus. Damit Sie während der Navigation im Serverobjektbaum jederzeit auf ein selektiertes Serverobjekt zugreifen können, weisen Sie der Tag-Eigenschaft des zuständigen *TreeNode* das erstellte Serverobjekt direkt zu. Damit erhalten Sie bei der Auswahl des Knotens einen direkten Zugriff auf den ausgewählten SQL Server. Jetzt ist es ein Leichtes, die Verbindung zu einem SQL Server aufzubauen (Listing 2 auf der Heft-CD). Ein paar Dinge zum Verbindungsaufbau sollten Sie aber beachten: Wenn Sie sich auf einem bereits verbundenen Server erneut verbinden, erhalten Sie eine *COM-Exception*. Das wäre nicht so schlimm, wenn Sie die Möglichkeiten hätten, den *Connect*-Status abzufragen – leider haben dies die Redmonder scheinbar vergessen. So bleibt Ihnen nichts weiter übrig, als es trotzdem einfach zu versuchen und auf eine *COM-Exception* vorbereitet zu sein. Dabei müssen Sie auf zwei Exceptions besonders achten:

```
-2147200991 SQL already loged in.
-2147201024 This Server object is already connected.
```

Auch das ist geschafft – ein paar wenige Zeilen Code und schon loggen Sie sich in jeden verfügbaren SQL Server ein. Den aufmerksamen Lesern unter Ihnen ist bestimmt eine kleine Unschönheit aufgefallen: *CloneDB* unterstützt nur *Windows Authentication*. Wollen Sie auch *SQL Server Authentication* unterstützen, brauchen Sie noch einen Dialog zur Eingabe von Benutzername und Passwort. Oder aber Sie nehmen den bequemen Weg: Der *SQL Server Enterprise Manager* ermöglicht das Registrieren von bevorzugten Servern und übernimmt das gesicherte Ablegen der Login-Informationen. Mit *SQL DMO* greifen Sie direkt auf diese Registrierungsinformationen zu. Die Collection *SQLDMO.Application.ServerGroups* hilft Ihnen, durch alle registrierten Servergruppen zu iterieren (Listing 3 auf Heft-CD).

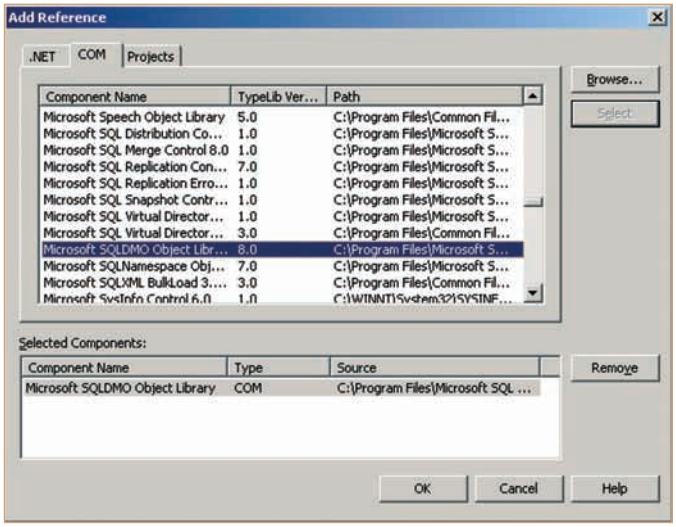

Abb. 2: SQL-DMO-Referenz dem Projekt hinzufügen

Die echten Highlights von SQL DMO

Wenn Sie Testdaten aus Ihrer Datenbank skripten möchten, brauchen Sie erst noch die Information, welche Datenbanken, Tabellen, *Views* und dergleichen sich überhaupt auf Ihrem Server befinden. Nichts leichter als das. Listing 4 (Heft-CD) zeigt Ihnen, wie Sie die Datenbanken Ihres SQL Server auflisten. Die Collection *SQLServer.Databases* gibt Ihnen für jede Benutzer- und Systemdatenbank ein *Database*-Objekt zurück. Es ist kaum anzunehmen, dass Sie sich auf jeden Server immer als Systemadministrator einloggen, daher sollten Sie Ihre Zugriffsberechtigung für jede einzelne Datenbank überprüfen. Genau dies macht die Zeile

```
if(myDB.UserProfile != SQLDMO_DBUSERPROFILE_TYPE.
                       SQLDMODbUserProf_InvalidLogin)
```

Sollten Sie keine Zugriffsberechtigung auf die Datenbank besitzen, wird *CloneDB* diese auch nicht auflisten. Natürlich können Sie Ihre Zugriffsberechtigung auch differenzierter abfragen. Da *CloneDB* aber nur Informationen ausliest und keinerlei

neue Objekte anlegen oder Wartungsarbeiten vornehmen möchte, reicht dieser einfache Test völlig. Im ersten Teil dieser Serie haben Sie von der Notwendigkeit der System Stored Procedure *sp_refreshview* gehört. Um diesen Aufruf erfolgreich durchzuführen, müssen Sie entweder *sysadmin*, *db_owner*, *db_ddlabmin* oder der *View Owner* sein. Aber nur wegen mangelnder Berechtigung für *sp_refreshview* auf die ganze Datenbank zu verzichten, wäre doch eine etwas zu große Einschränkung. Wie war das noch mit dem ungelösten, ultimativen Problem aus Teil 1? Genau – die Frage lautete „Wie kommen Sie am einfachsten an Ihre *CREATE-TABLE*-Skripts?“. Mal abgesehen von ein paar Zeilen Code zur Kosmetik (Listing 5 auf Heft-CD), ist das ganze Problem mit nur einer Zeile gelöst:

```
DBObject.Srcipt([ ScriptType ] [, ScriptFilePath ]
                                        [, Script2Type ])
```

Auf diese ganz einfache Weise lässt sich für jedes erdenkliche Datenbankobjekt ein Skript erzeugen. Angefangen bei Tabellen und Stored Procedures bis hin zu Logins und Jobs. Wie mächtig die Skriptvariante ist, erahnen Sie, wenn Sie einen Blick auf die Konstantenauflistung werfen (z.B. im Rahmen des Objekt-Browsers). Übrigens, wenn Sie sich mit dem SQL Server Enterprise Manager ein SQL-Skript generieren lassen, erhalten Sie die genau gleichen Skripts. Kein Wunder, benutzt doch auch der SQL Server Enterprise Manager ausgiebig *SQL DMO*. Wenn Sie die in diesem Artikel vorgestellten Konzepte umsetzen, die T-SQL-Befehle aus dem ersten Teil einfließen lassen und noch ein wenig Kosmetik anbringen, haben Sie *CloneDB* fertig implementiert. Jetzt hat *CloneDB* einen Stand erreicht, mit dem Sie die ersten Testdaten auf dem Produktivsystem skripten und auf Ihrer Testdatenbank anwenden können. Sie werden sehen: Das klappt schon recht gut und Sie sparen mit *CloneDB* nicht nur viel Zeit – nein, endlich wird sich auch Ihre Testdatenbank mit realen Testdaten füllen.

Nach *DMO* kommt *SMO*

Die ersten Bewährungsproben hat *CloneDB* längst hinter sich – es hat sich in realen Projekten bereits bestens bewährt. Natürlich fehlt noch das eine oder andere Feature – aber der Sourcecode ist ja vorhanden und so sind neue Anforderungen schnell implementiert. Was vielmehr stört, ist der Einsatz von *SQL DMO* als COM-API in einer .NET-Applikation. Nein, es ist mehr als nur der (falsche?) Stolz des Autors, eine pure .NET-Applikation schreiben zu wollen. *SQL DMO* passt einfach nicht so recht in die .NET-Welt. So ist zum Beispiel die *SQL-DMO-Connection* nicht kompatibel zu einer *ADO.NET-Connection*, oder das *SQL-DMO-QueryResult* als Ergebnis einer SQL-Abfrage passt nicht in eine *ADO .NET-DataTable* – von der Inkompatibilität der SQL-DMO-Datentypen mit denen von .NET ganz zu schweigen. Aber auch hierfür gibt es ein Zauberwort – *SMO*. *SMO* steht für *Server Management Objects* und wird zusammen mit SQL Server 2005 ausgeliefert. *SMO* ist pures .NET und ersetzt *SQL DMO*. In einer der nächsten Ausgaben des *dot.net magazin* erfahren Sie, wie Sie *CloneDB* dank *SMO* effizient einsetzen und damit kein COM mehr benötigen. Einverstanden – lediglich dieselbe Funktion erneut zu implementieren ist nicht sehr spannend. Wie wäre es damit: Jedes T-SQL-Kommando soll vor dem Senden an den Server angezeigt werden und die Möglichkeit bieten, dieses von Hand abzuändern. Das gilt für alle T-SQL-Kommandos, inklusive denjenigen, die *SMO* selber an den SQL Server sendet. Eine solche Funktion besitzt gleich zwei Nutzen: Zum einen lernen Sie so eine Menge über die Funktionsweise von *SMO* und zum anderen erhalten Sie die Möglichkeit, die Funktionsweise von *CloneDB* für Spezialfälle anzupassen, ohne dass Sie sich jedes Mal gleich hinter die Source machen müssen. Wagen Sie sich doch schon mal selber an dieses interessante Problem ran. Brauchen Sie eine kleine Starthilfe? Schreiben Sie mir ein E-Mail und ich sende Ihnen ein Code-Snippet. ●

Urs Gehrig ist Head of System Services der bbv Software Services AG. Dieser Service hat sich auf die kundenorientierte Softwareentwicklung mit .NET spezialisiert. Er selbst ist seit 1989 als Entwickler von Windows-Applikationen und EDV-Berater tätig. Seine Steckenpferde sind Datenbanken und Security. Sie erreichen Ihn unter urs.gehrig@bbv.ch.

SQL DMO – ein starkes COM-API für SQL Server

SQL Distributed Management Objects (*SQL DMO*) ist eine Sammlung von Objekten für Management- und Replikationsaufgaben mit dem SQL Server und bietet Support für

- administrative Aufgaben mit dem SQL Server, z.B. Backup von Datenbanken oder Starten eines Servers.
- das Anlegen und Administrieren von SQL-Server-Objekten, z.B. Löschen von Tabellen oder Zuweisen von Zugriffsberechtigungen für *Views*.
- das Anlegen und Administrieren von SQL Server Agent Jobs, Alerts und Operatoren.
- das Installieren und Konfigurieren von SQL-Server-Replikation.

Viele Objekte sind in einer einfachen wie auch erweiterten Version verfügbar, z.B. *Table* und *Table2*, *SQLServer* und *SQLServer2*. Die erweiterten Versionen (Suffix 2) enthalten zusätzliche Methoden und Attribute, welche nur vom SQL Server 2000 unterstützt werden. *SQL-DMO*-Applikationen lassen sich sowohl für den Server als auch den Client entwickeln. Der Verbindungsaufbau zum SQL Server wie auch die Kommunikation mit diesem erfolgt immer via ODBC. Für SQL Server 7.0 und 2000 ist *SQL DMO* in *SQLDMO.DLL* und dem Ressourcen-File *SQLDMO .RLL* implementiert. Für einen englischen SQL Server 2000 finden Sie diese beiden Dateien standardmäßig in den Verzeichnissen *C:\Program Files\Microsoft SQL Server\80\Tools\Binn* und *C:\Program Files\Microsoft SQL Server\80\Tools\Binn\resources\1033*. Bevor Sie *SQL DMO* verwenden, müssen Sie sicherstellen, dass Sie die beiden Dateien mittels *RegSvr32* registriert haben. Als *Dual Interface COM* und *In-Process*-Server implementiert, ist *SQL DMO* auch in C# verwendbar.

Links & Literatur

[1] SQL DMO in MSDN: msdn.microsoft.com/library/default.asp?url=/library/en-us/sqldmo/dmoref_con01_2yi7.asp

[2] SQL Server Developer Center: msdn.microsoft.com/sql/

[3] Urs Gehrig: Aller Anfang ist schwer, in: *dot.net magazin* 9.2005

[4] Urs Gehrig: Optimale Performance mit SQL Server 2000, in: *dot.net magazin* 3/4.2004