HAUFE.
Ausgabe 05/2018

ACCESS IM UNTERNEHMEN

SQL SERVER

Der SQL Server – das Schwerpunktthema in diesem Heft! Alles rund um die Themen Management Studio, Benutzerverwaltung und Sicherheit.

In diesem Heft:

SICHERHEIT UND BENUTZERVERWALTUNG

Schützen Sie die Daten mit dem den Sicherheitsfunktionen des SQL Servers.

SEITE 31

ZUGRIFFE UNTERSUCHEN MIT XEVENTS

Überwachen Sie Datenverkehr im SQL Server mit den erweiterten Eigenschaften.

SEITE 68

ODBCDIRECT DURCH DIE HINTERTÜR

Greifen Sie mit einer Alternative zu ADODB auf SQL Server und andere RDBMS zu.

SEITE 2

SQL Server

Lange haben wir das Thema SQL Server eher stiefmütterlich behandelt. Das soll sich nun ändern: In der aktuellen Ausgabe schaffen wir einige Grundlagen, in denen es sich vor allem um den SQL Server, das SQL Server Management Studio, die Verwaltung von Datenbanken und um das Sicherheitssystem geht. In den folgenden Ausgaben starten wir dann durch und schauen uns an, wie Sie Anwendungen auf Basis eines Access-Frontends mit SQL Server Backend programmieren können.

Wir starten mit dem Beitrag **SQL Server Management Studio**, wo wir die grundlegenden und für die folgenden Beiträge wichtigen Funktionen vorstellen – zum Beispiel die Anmeldung an einer SQL Server-Datenbank (ab S. 12).

Damit Sie mit den Beispieldatenbanken, die wir in den Beiträgen dieser Ausgabe und der folgenden Ausgaben etwas anfangen können, zeigen wir ab S. 23 unter dem Titel **SQL Server-Datenbanken kopieren**, wie Sie mit verschiedenen Techniken eine Datenbank auf dem einen SQL Server exportieren und im anderen SQL Server als neue Datenbank anlegen können. Die beiden Methoden sind das Erstellen und Wiederherstellen einer Datenbanksicherung sowie das Exportieren der kompletten Datenbank in Form eines SQL-Skripts, dass wir auf einem anderen SQL Server ausführen und damit eine neue Datenbank mit dem Inhalt der exportierten Datenbank erstellen.

Der Beitrag **SQL Server: Sicherheit und Benutzerverwaltung** zeigt ab S. 31 im Detail, welche Authentifizierungsarten es gibt, wie Sie sich über die Windows- und die SQL Server-Authentifizierung anmelden, wie Sie Anmeldungen für Windows-Benutzer oder -Benutzergruppen hinzufügen und für diese den Zugriff auf einzelne Datenbanken freischalten. Dabei gehen wir zu Beispielzwecken auch noch auf das Anlegen von Benutzergruppen und Benutzern per Befehlszeilentool ein. Schließlich setzen wir noch Berechtigungen für die verschiedenen Elemente einer Datenbank.

Ab Seite 47 gehen wir im Beitrag **SQL Server-Authentifizierung** auf SQL Server Authentifizierung ein. Diese bietet gegenüber der Windows-Authentifizierung einige Besonderheit, die Sie kennen müssen – vor allem, wenn Ihre Umgebung nicht die Nutzung der Windows-Authentifizierung erlaubt, sondern nur die der SQL Server-Authentifizierung.

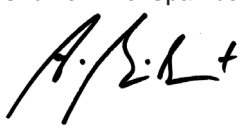
In einem Beitrag wollen wir zumindest kurz auf den Zugriff auf eine SQL Server-Datenbank von Access aus eingehen, und zwar unter dem Titel **SQL Server-Zugriff ohne gespeichertes Kennwort** (ab S. 52). Hier zeigen wir, wie Sie sie eine sichere Verknüpfung von Datenbanken unter Access erreichen!

Schließlich testen wir noch die verschiedenen Authentifizierungsmethoden mit verschiedenen Benutzerkonten – siehe ab Seite 64 im Beitrag **Authentifizierung im SQL Server testen**.

Was überhaupt alles im SQL Server passiert, wenn Sie beispielsweise von Access aus auf eine Datenbank zugreifen, können sie mit den XEvents ansehen. Diese stellen wir im Beitrag **SQL Server: Zugriffe untersuchen mit XEvents** ab S. 68 vor.

Schließlich finden Sie ab Seite 2 unter dem Titel **ODBC-Direct durch die Hintertür** noch einen Beitrag, der sich nicht ausschließlich um den SQL Server dreht, sondern den Zugriff auf RDBMS im Allgemeinen mit der ODBCDirect-Schnittstelle beschäftigt.

Und nun: Viel Spaß beim Lesen!



Ihr André Minhorst

ODBCDirect durch die Hintertür

Haben Sie ihr Backend in ein DBMS ausgelagert, also die Tabellen auf einen SQL-Server migriert, so steht Ihnen ab Access 2007 nur noch die Schnittstelle ADODB zur Verfügung, um unter VBA auf sie zuzugreifen, falls Sie sich nicht nur auf verknüpfte Tabellen beschränken möchten. Denn die Technologie ODBCDirect, mit der man direkt eine Verbindung zum Server aufbauen konnte, wurde ersatzlos gestrichen. Mit einem Trick schlagen Sie Microsoft jedoch ein Schnippchen!

ODBCDirect

Es handelt sich dabei eine Datenschnittstelle, die in der Bibliothek **DAO** untergebracht ist. Dort finden Sie ein **Connection**-Objekt, welches allein für die Kommunikation mit einem SQL-Server über **ODBC** zuständig ist (Bild 1). Doch offenbar enthält die Bibliothek diese Klasse nur noch aus Kompatibilitätsgründen, denn jeglicher Methodenaufruf auf sie endet mit einer Fehlermeldung 3827: **ODBCDirect wird nicht mehr unterstützt. Schreiben Sie den Code so um, dass ADO anstelle von DAO verwendet wird**. Das können Sie tun, denn etwas anderes bleibt Ihnen nun auch nicht übrig.

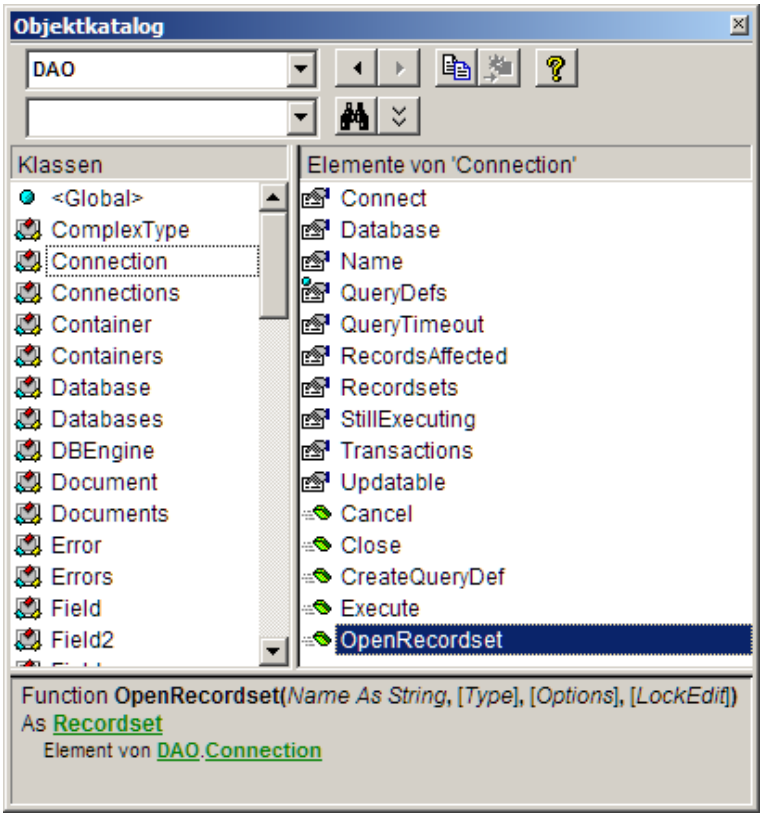


Bild 1: Die **ODBCDirect**-Klasse **Connection** im Objektkatalog von VBA

Der Sachverhalt stellt sich in der Praxis nicht selten ein. Haben Sie etwa eine ältere **Access 2003**-Datenbank im Einsatz, die **ODBCDirect** nutzt, und auf den Rechnern soll **Office 2010** installiert werden, so kommen Sie um Anpassungen leider nicht herum. Statt der **DAO-Connection** verwenden Sie nun überall eine **ADODB-Connection**, deren Methoden aber deutlich abweichen! Hier ist einiges an Arbeit für die Migration angesagt.

Warum Microsoft die Schnittstelle entfernte, ist nicht nachzuvollziehen. Einerseits wurden die sogenannten **Access-Projekte** (**ADP**) in welchen Sie Formulare unmittelbar an einen **MSSQL**-Server binden konnten, ab **Access 2007** komplett entfernt und stattdessen wurden ODBC-verknüpfte Tabellen empfohlen. Gleichzeitig aber strich man **ODBCDirect**. Ich habe dafür nur eine Erklärung: Einst waren SQL-Server ziemlich teuer, weshalb sie häufig für Low Budget-Lösungen mit Access nicht infrage kamen. Zudem scheuten viele Entwickler den Einsatz des SQL-Servers, weil hier eine Menge zusätzliches Knowhow zu erwerben war. In der Folge fanden Sie in den einschlägigen Foren und Newsgroups nur selten Fragen zum Thema **ODBCDirect**. Für Microsoft offenbar Grund genug, sich dieser Schnittstelle zu entledigen.

Inzwischen aber stellt nicht nur Microsoft selbst einen kostenlosen leistungsfähigen Server (**SQL Server Express**) zur Verfügung, sondern auch andere Größen der Branche, wie **IBM** mit **DB2 Express** oder **Oracle** mit E**xpress 11g**, von den **OpenSource**-Lösungen **MySQL**, **MariaDB**, **Postgres** und **Firebird** ganz zu schweigen, sodass der Einsatz dieser **DBMS** als Backend, die der A**ccess Engine** klar den Rang ablaufen, naheliegt.

Wollen Sie hier **ADODB** verwenden, so hat dies einen Haken: Viele der genannten **DBMS** stellen keinen **OLEDB-Provider** zur Verfügung. Manche sind instabil oder wenig performant, bei manchen finden Sie gar keinen. Dann müssen Sie den **ODBC-Provider MSDASQL** nehmen, der aber eine zusätzliche Ebene einzieht und deshalb in der Performance das Nachsehen hat.

Empfehlung: Voten Sie auf **access.uservoice** für die Wiederaufnahme von **ODBCDirect**! Dort nimmt Microsoft Vorschläge zu zukünftigen Access-Versionen entgegen. Früher einmal gab es das **Connect**-Programm von Microsoft, wo vornehmlich **MVPs** ihre Vorschläge einbrachten. Die wurden allerdings nicht wirklich ernst genommen. Offenbar hat sich dies geändert. Nach dem Debakel mit den **Access-WebServices**, die in der Zukunft deshalb eben entfallen sollen, wendet man sich nun anscheinend vermehrt den Anregungen der Community zu, wie etwa die Wiedergeburt der **dBase**-Schnittstelle zeigt.

Die Programmierung der **ODBCDirect**-Schnittstelle ist fast so einfach, wie die von lokalen oder verlinkten Access-Tabellen:

```
Dim oWrk As DAO.Workspace
Dim oConx As DAO.Connection
Dim sConx As String
Dim rs As DAO.Recordset

Set oWrk = oDbEng.CreateWorkspace("", "", "", dbUseODBC)
sConx = "ODBC;Driver={Microsoft dBase Driver (*.dbf)}" & _
        ";DriverID=533";Dbq=" & CurrentProject.Path
Set oConx = oWrk.OpenConnection("", , , sConx)
Set rs = oConx.OpenRecordset( _
    "SELECT * FROM KUNDENLI.DBF.DBF", dbOpenDynaset)
```

Hier wird zunächst ein Workspace **oWrk** angelegt, welches von dem abweicht, das die **DBEngine** von **Access.Application** voreingestellt hat, weil die Konstante **dbUseODBC** angegeben ist. Der Normalfall wäre **dbUseJet**. Dann wird in **sConx** der **Connect**-String zusammengebaut, der mit dem Tag **ODBC;** beginnen muss.

Mit **Driver** geben Sie den Namen eines installierten ODBC-Treibers an, in unserem Fall den **dBase**-Treiber. Daran schließen sich treiberspezifische Optionen an. Hier ist es vor allem der Pfad, unter dem sich die **dBase**-Dateien befinden, im Schlüssel **Dbq**.

Mit diesem **Connect**-String öffnen Sie das **Connection**-Objekt **oConx**. Ähnlich, wie bei einem **Database**-Objekt, können Sie nun ein **Recordset** über ein SQL-Statement auf die **Connection** öffnen.

Aber das geht doch gar nicht mehr!
Korrekt. Führen Sie den angeführten Code in einer **ACCDB** aus, so tritt die erwähnte Fehlermeldung auf.

Sie könnten nun auf die Idee kommen, statt der **ACE**-Bibliothek die alte zu verwenden. Löschen Sie also den Verweis auf **DAO** und laden Sie einen auf **DAO.36** neu (**Microsoft DAO 3.6 Object Library**). Tatsächlich ändert dies am Sachverhalt rein gar nichts. Denn der Verweis führt nicht dazu, dass die alte Engine geladen würde.

Da die alte, wie die neue, den Namen **DAO** trägt, verwendet Access weiterhin die interne **ADEDAO.dll**, die bereits beim Start von Access geladen wird. Die Bibliothek selbst wird zwar von VBA genutzt, doch quasi nur als Tabelle für die Klassen, die unter beiden Bibliotheken an dieselben **GUIDs** und Methoden für die **Interfaces** gebunden sind.

Der nächste Versuch wäre, eine eigene **DBEngine** anzulegen, die auf die **dao360.dll** verweist. Das lässt sich über die Methode **CreateObject** erledigen:

```
Dim oDBEng As Object
Dim oWrk As Object
Set oDbEng = CreateObject("DAO.DBEngine.36")
Set oWrk = oDbEng.CreateWorkspace("", "", "", dbUseODBC)
```

Verblüffend, auch dies zeitigt dieselbe Fehlermeldung in der letzten Zeile! Hier ist der Grund der, dass die Office-Installation in der Registry die ProgID- und CLSID-Schlüssel so verbogen hat, dass sie ebenfalls auf die acedao.dll verweisen. Also scheinen wohl alle Wege verbaut zu sein.

Es geht doch!

In der letzten Ausgabe erfuhren Sie, wie Sie **COM-Objekte** instanziieren können, ohne dass jene im System registriert sein müssen (**ActiveX und COM ohne Registrierung verwenden**). Dabei kommt ein Modul zum Einsatz (**mdlComLoaderASM**), das ziemlich komplex ist und auch vor **Assembler**-Teilen nicht Halt macht. Hier haben Sie nun eine nützliche Anwendung für die Routinen des Moduls: Wir versuchen, eine Instanz der alten **JET-Engine** über die Funktion **GetCOMInstance** zu erhalten. Das geht im Prinzip so:

```
Dim oDbEng As DAO.DBEngine
Dim oWrk As DAO.Workspace
Set oDbEng = GetCOMInstance(sFile, "DBEngine", True)
Set oWrk = oDbEng.CreateWorkspace("", "", "", dbUseODBC)
```

Und, siehe da, die Sache klappt, und wir haben Microsoft ausgetrickst! Natürlich muss in der Variablen **sFile** zuvor der Pfad zur dao360.dll stehen.

Aber gibt es diese Datei überhaupt auf dem Rechner mit **Office 2007** ff.? Wir können das nicht für alle Fälle versichern. Doch die **dao360.dll** wird auch von Windows selbst für seine Zwecke genutzt, weshalb die Datei nach unseren Beobachtungen mit diesem installiert wird. Auch der Pfad scheint immer derselbe zu sein. Sie können ihn vom Pfad der **ACEDAO** ableiten, wenn auf diese ein Verweis gesetzt ist:

```
sFile = References("DAO").FullPath
sFile = Replace(sFile, "OFFICE14\ACEDAO.DLL", _
                       "DAO\dao360.dll")
```

Heraus kommt dabei der String

```
C:\Program Files (x86)\Common Files\Microsoft Shared\DAO\
dao360.dll
```

Möchten Sie also Ihre alte **Access 2003**-Datenbank, die **ODBCDirect** nutzt, nach **Access 2010** migrieren, so ersetzen Sie im Code einfach alle Aufrufe auf das **Connection**-Objekt etwa mit dieser Routine:

```
Dim oConx As DAO.Connection
Dim oDbEng As DAO.DBEngine
Dim oWrk As DAO.Workspace

Function Connection As DAO.Connection
Dim sFile As String
If oConx Is Nothing Then
  sFile = References("DAO").FullPath
  sFile = Replace(sFile, "OFFICE14\ACEDAO.DLL", _
                         "DAO\dao360.dll")
  Set oDbEng = GetCOMInstance(sFile, "DBEngine", True)
  Set oWrk = oDbEng.CreateWorkspace("", "", "", dbUseODBC)
End If
Set Connection = oConx
End Function
```

Das **Connection**-Objekt steht damit generell im VBA-Projekt zur Verfügung. Auf Umgestaltung nach **ADO** können Sie getrost verzichten.

Ein Anwendungsbeispiel finden Sie in Listing 1. Wie zuvor ausgeführt erzeugt **GetCOMInstance** hier eine

neue **DBEngine**. Statt der **ProgID DBEngine** allerdings ist die entsprechende **CLSID** für das Objekt der Begierde eingesetzt. **GetCOMInstance** muss nämlich sonst die **CLSID** aus der **ProgID** selbst ermitteln, was Sie der Prozedur abnehmen können.

Den **Connect**-String **sConx** verweist in diesem Beispiel auf einen **MySQL-ODBC**-Treiber. Der Server wird über den Standard-Port **3306** auf die lokale Maschine mit der **IP 127.0.0.1** angesprochen und der Benutzername zur Autorisierung am Server ist **root**, das Passwort ist **aiupwd**. Die **Connection oConx** wird über diesen String geöffnet.

Interessant ist in diesem Zusammenhang die Konstante **dbDriverNoPrompt**. Sie sagt **ODBCDirect**, dass kein Dialog erscheinen soll, wenn etwas am **Connect**-String nicht stimmt. In diesem Fall käme es zu einer Fehlermeldung.

```
Sub ConnectMysqlODBCDirect()
    Dim oDbEng As DAO.DBEngine
    Dim oWrk As DAO.Workspace
    Dim oConx As DAO.Connection
    Dim sConx As String
    Dim rs As DAO.Recordset
    Dim sFile As String

    sFile = Replace(References("DAO").FullPath, "OFFICE14\ACEDAO.DLL", "DAO\dao360.dll")

    Set oDbEng = GetCOMInstance(sFile, _
                    "{00000100-0000-0010-8000-00AA006D2EA4}", True)
    'oder statt CLSID: DBEngine ~ DAO.DBEngine.36

    Set oWrk = oDbEng.CreateWorkspace("", "", "", dbUseODBC)
    sConx = "ODBC;Driver={MySQL ODBC 5.3 ANSI Driver};" & _
        "DATABASE=mysqlbackend;" & _
        "SERVER=127.0.0.1;PORT=3306;" & _
        "UID=root;PWD=aiupwd;OPTION=18475"
    Set oConx = oWrk.OpenConnection("", dbDriverNoPrompt, , sConx)
    'oder dbDriverCompleteRequired
    Set rs = oConx.OpenRecordset( _
      "SELECT ID,Artikel FROM tblArtikel", dbOpenDynaset)
    Do While Not rs.EOF
        Debug.Print rs(0).Value, rs(1).Value
        rs.MoveNext
    Loop
    rs.Close

    On Error Resume Next
    oWrk.Close
    Set oWrk = Nothing
    oConx.Close
    Set oConx = Nothing
    Set oDbEng = Nothing
End Function
```

Listing 1: Diese Prozedur erzeugt eine Instanz der **DBEngine** und spricht damit einen **MySQL**-Server an.

Setzten Sie stattdessen **dbDriverCompleteRequired** ein, so erscheint der **ODBC**-Dialog des **Datenquellenadministrators** von Windows. In dem sind dann die korrekten Angaben zur Verbindung vorausgefüllt. Lassen Sie etwa den Benutzernamen weg, so können Sie diesen dann im Dialog nachreichen.

Schließlich öffnet sich ein Recordset **rs** auf die Tabelle **tblArtikel** auf die **Connection**. In der folgenden Schleife werden dann die Werte der Felder **ID** und **Artikel** der Datensätze alle im VBA-Direktfenster ausgegeben. Zu erwähnen wäre, dass diese Methode erheblich performanter ist, als **ADO** über **ODBC** auf den MySQL-Treiber. Es gibt nämlich keinen funktionierenden kostenlosen **MySQL**- oder **MariaDB-OLEDB-Provider**.

dBase über ODBCDirect

Noch ein Feature ist ab **Access 2013** dem Rotstift anheimgefallen: die Schnittstelle zu **dBase**. Bis dahin konnte man noch über das **Ribbon** und **Externe Daten dBase**-Dateien sowohl importieren, wie verknüpfen oder exportieren. Das geht unter Umständen eben nicht mehr, obwohl die Schnittstelle im neuesten **Access 2016** wieder drin ist. **dBase** ist als Austauschformat und Schnittstelle zu manchen anderen Programmen (leider) noch immer aktuell.

Der Umweg über eine **DSN**, welche Sie auf **dBase**-Dateien im **ODBC-Datenquellen-Administrator** anlegen, klappt ebenfalls nicht. Der Versuch, darauf eine **ODBC**-Verknüpfung in Access aufzubauen scheitert mit der Meldung **Verknüpfen von externen ISAM-Datenbanktabellen mit Ihrer Datenbank über ODBC ist nicht möglich** – warum auch immer das so unsinnigerweise vorgesehen wurde. Die einzige Lösung ist hier in VBA **ODBCDirect** im Verein mit **GetCOMInstance**. Auch eine **Passthrough-Abfrage** auf den **ODBC**-Treiber funktioniert nicht, wie die Abfrage **qry_PT_dBase** der Demodatenbank zeigt.

Kleiner Tipp am Rande: Unter **Windows 64** existiert wohl schon im Startmenü ein Link zum **ODBC-Datenquellenadministrator**. Doch dieser öffnet die 64bit-Version desselben, in der demnach auch nur 64bit-Treiber aufgelistet sind. Mit diesen kann **Office x32** nichts anfangen. Legen Sie sich deshalb eine Verknüpfung zur Version unter **SysWOW64** auf dem Desktop an (**C:\Windows\SysWOW64\odbcad32.exe**). Sie sollte nach dem Start den Dialog ähnlich zeigen, wie in Bild 2.

Sie können nun eine neue **Benutzer-DSN** zu **dBase**-Dateien anlegen, indem Sie unter dem entsprechenden

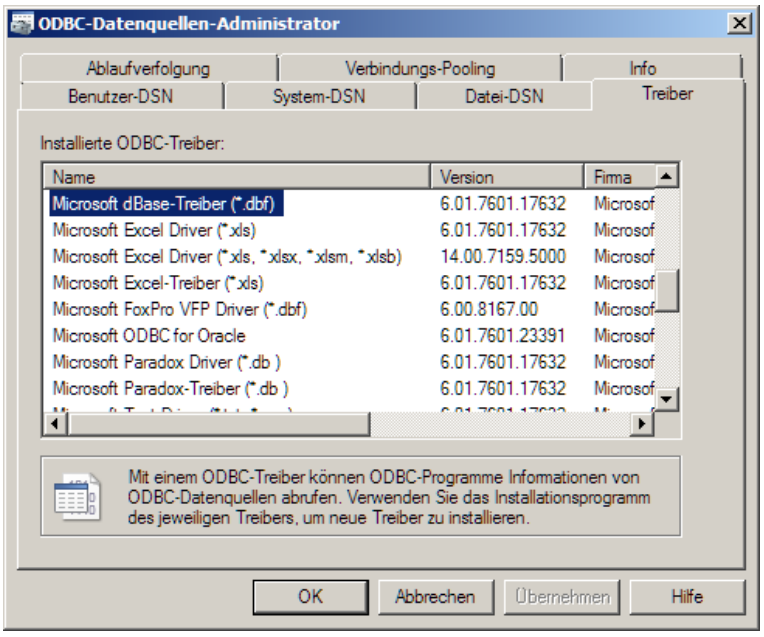


Bild 2: Der 32bit-**ODBC-Datenquellen-Administrator** listet die 32bit-Treiber auf.

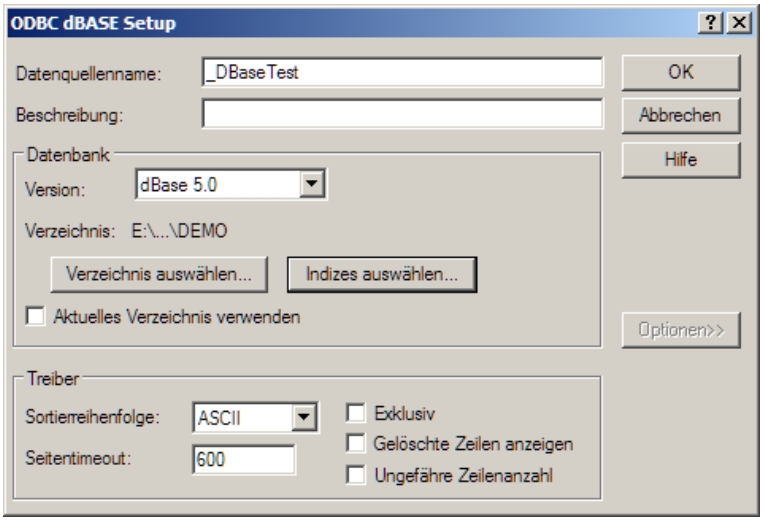


Bild 3: So sieht die Anlage einer **DSN** im **ODBC-Dialog** zu **dBase**-Dateien aus.

Reiter die Schaltfläche Neu anklicken. In Bild 3 ist der Dialog nach Auswahl des **dBase**-Treibers ausgefüllt. Übrigens können wir auch nicht garantieren, dass der Treiber auch wirklich auf Ihrem System installiert ist. Im Zweifel können Sie das nachholen, indem Sie das **Microsoft ODBC Desktop Driver Pack** installieren. Infos zum **dBase**-Treiber gibt es dann etwa in den **Microsoft Docs**.

Für unsere **ODBCDirect**-Verbindung benötigen wir eine solche **DSN** nicht, weil wir die **DSNless** anlegen. Der Dialog macht jedoch deutlich, was es mit dem **dBase-DBMS** auf sich hat. Sie wählen am besten die Treiberversion **5.0** aus, die im **Connect**-String dem Eintrag **DriverID=533** entspricht. Ansonsten geben Sie lediglich ein Verzeichnis an, in dem sich die **dBase**-Dateien befinden, was sich im Parameter **Dbq** wiederspiegelt. Die Verbindung geschieht also nicht zu einer einzelnen Datei, sondern zu einem Verzeichnis, das quasi die Datenbank darstellt, während die Dateien die einzelnen Tabellen sind.

```
Sub ConnectdBaseODBCDirect()
    Dim sFile As String, sPath As String
    Dim oDbEng As DAO.DBEngine
    Dim oWrk As DAO.Workspace
    Dim oConx As DAO.Connection, sConx As String
    Dim rs As DAO.Recordset, fld As DAO.Field

    sPath = CurrentProject.Path
    sFile = References("DAO").FullPath
    sFile = Replace(sFile, "OFFICE14\ACEDAO.DLL", "DAO\dao360.dll")
    Debug.Print sFile

    Set oDbEng = GetCOMInstance(sFile, _
        "{00000100-0000-0010-8000-00AA006D2EA4}", True)
    Set oWrk = oDbEng.CreateWorkspace("", "", "", dbUseODBC)
    sConx = "ODBC;Driver={Microsoft dBase Driver (*.dbf)};DriverID=533;Dbq=" & sPath
    Set oConx = oWrk.OpenConnection("", dbDriverCompleteRequired, , sConx)
    Set rs = oConx.OpenRecordset("SELECT * FROM KUNDENLI.DBF", dbOpenDynaset)
    For Each fld In rs.Fields
        Debug.Print fld.Name,
    Next fld
    Debug.Print
    Do While Not rs.EOF
        For Each fld In rs.Fields
            Debug.Print fld.Value,
        Next fld
        Debug.Print
        rs.MoveNext
    Loop
    rs.Close

    On Error Resume Next
    oWrk.Close: Set oWrk = Nothing
    oConx.Close: Set oConx = Nothing
    Set oDbEng = Nothing
End Function
```

Listing 2: Die Prozedur öffnet eine **dBase**-Tabelle und gibt sämtliche Datensätze im Direktfenster aus.

Listing 2 zeigt den kompletten Code, der an sich nur eine erweiterte Version der **MySQL**-Routine darstellt. Interessant ist hier die SQL-Syntax beim Öffnen des Recordset. Da eine **dBase**-Datei imgrunde eine Tabelle darstellt, müssen Sie die auch im **FROM**-Statement samt Dateierweiterung angeben (**KUNDENLI.dbf**).

Die Routine schreibt über die erste Spalte alle Feldnamen der Tabelle in das Direktfenster. In der zweiten verschachtelten Schleife geschieht dies jeweils für jeden Datensatz, wobei dann nicht **Field.Name** ausgegeben wird, sondern **Field.Value**. Die **dBase**-Datei ist durch Export aus einer Access-Tabelle **Kundenliste** entstanden. Sie sehen, dass **dBase** den Namen abkürzt.

Unter **dBase III** können Tabellennamen nur acht Zeichen lang sein. Feldnamen können länger sein. Weitere Spezifikationen finden Sie etwa unter folgendem Link:

http://www.manmrk.net/tutorials/database/xbase/dbase_spec.html

ODBCDirect-Daten im Formular

Zwar können Sie nun unter VBA über **ODBCDirect** auf die Daten eines SQL-Servers zugreifen. Vielleicht aber möchten Sie diese auch in einem Formular anzeigen, anstatt sie lediglich per Code weiterzuverarbeiten.

Über die Eigenschaft **Recordset** eines Formulars, die nicht nur Lese-, sondern auch Schreibzugriff gestattet, könnten Sie ein **ODBCDirect**-Recordset direkt zuweisen. Das allerdings funktioniert nicht. Sie erhalten bei Zuweisung die Fehlermeldung **Das Objekt ist kein gültiges Recordset.** Denn Formulare unterstützen nur die normalen **DAO**-Recordsets oder **ADO**-Recordsets.

Entweder verzichten Sie nun auf Formularfelder und geben die Feldwerte des Recordsets in einem Listenfeld aus, oder Sie wandeln das **ODBC-Recordset** in ein anderes vom Formular unterstütztes um. **DAO** scheidet hier eigentlich aus, weil Sie da ein **Recordset** nur auf eine physische Tabelle öffnen können. Anders bei **ADO**, wo ein **Disconnected Recordset** auch künstlich erzeugt werden kann. Zu dieser Methode haben wir gegriffen, wobei die Lösung nur eine rudimentäre ist, da auf Datentypkonvertierung verzichtet wurde. Der Kern der Lösung besteht in der Klasse **cODBCConnection**, welche **ODBCDirect** komplett kapselt.

cODBCConnection hat folgende Methoden und Eigenschaften:

- Im Property **ConnectString** geben Sie die Verbindungzeichenfolge an oder lesen sie aus.
- **ConnectSilent** schaltet, falls True, den ODBC-Dialog aus, wenn im Verbindungs-String etwas nicht stimmt.
- **DAODLL** gibt rein informativ als String den Ort der **dao360.dll** aus.
- Die Funktion **Connect** stellt dann erst die Verbindung her und gibt ein **DAO.Connection**-Objekt zurück.
- Der Methode **OpenRecordset** übergeben Sie einen SQL-String, wie gewöhnt. Als Ergebnis erhalten Sie ein **DAO-ODBC-Recordset**.
- **OpenRecordsetADO** tut dasselbe, gibt aber direkt ein **ADODB-Recordset** zurück, welches Sie direkt einem Formular zuweisen können.

Die letzte Methode ist auch die einzige, die wir näher beleuchten wollen, da die anderen Routinen weitgehend nur das bisher schon Gesagte wiederspiegeln. Listing 3 stellt die Prozedur dar.

Die Objektvariable **rs** speichert zunächst das ODBC-Recordset, welches über die andere Klassenfunktion **OpenRecordset** und den übergebenen SQL-String erhalten wird. Anschließend wird ein mit **New** deklariertes **ADODB-Recordset** von Grund auf neu erstellt. Dazu klappert die erste Schleife über die **Fields**-Auflistung alle Felder des **ODBC-Recordsets** ab und fügt dem ADO-Recordset **rsADO** gleichnamige hinzu, wobei als Datentyp **adBSTR** verwendet wird. Das ist der String-Typ für normalen Text. **adVarChar** scheidet aus, weil Access hier nicht mit Unicode-Zeichen klarkäme. Nach dem Öffnen des ADO-Recordsets per **Open**-Anweisung können diesem direkt per **AddNew** neue Datensätze angefügt werden, was die zweite Schleife erledigt. Ein Update nach jedem Durchgang ist bei **ADO** nicht nötig: Das abschließende **UpdateBatch** speichert alles auf einen Schlag, wie bei einer Transaktion. Das **ODBC-Recordset rs** kann nun geschlossen werden. Das Recordset **rsADO** wird nun als Rückgabewert der Funktion gesetzt.

So kurz die Routine auch ist, die Performance ist durch diese Konvertierung natürlich nicht überwältigend. Zur Anzeige in einem Formular spielt dies jedoch eine untergeordnete Rolle. Klar ist zudem, dass die Daten dieses Recordsets zwar auch beschrieben oder neue hinzuge-

```
Function OpenRecordsetADO(ByVal sSQL As String) _
                                    As ADODB.Recordset
    Dim rs As DAO.Recordset,.
    Dim fld As DAO.Field
    Dim rsADO As New ADODB.Recordset

    Set rs = OpenRecordset(sSQL)
    rsADO.CursorLocation = adUseClient
    For Each fld In rs.Fields
        rsADO.Fields.Append fld.Name, adBSTR
    Next fld
    rsADO.Open
    rs.MoveFirst
    Do While Not rs.EOF
        rsADO.AddNew
        For Each fld In rs.Fields
            rsADO(fld.Name).Value = Nz(fld.Value)
        Next fld
        rs.MoveNext
    Loop
    rsADO.UpdateBatch adAffectAll
    rs.Close
    Set OpenRecordsetADO = rsADO
End Function
```

Listing 3: Die Funktion konvertiert ein **ODBC**-in ein **ADO-Recordset**.

fügt werden können, dies sich jedoch nicht rückwärts auf die **dBase**-Datei auswirkt.

Diese Klasse **cODBCConnection** wird nun vom Formular **frmKundenliste** der Demo benutzt, um seine Textfelder zu füllen. Der Entwurf des Formulars in Bild 4 zeigt zehn Textfelder und zehn an sie geknüpfte Labels. Die Ansicht ist auf **Datenblatt** eingestellt, weshalb der Inhalt der Labels später die Spaltenköpfe angibt. Die Labels haben Namen von **L1** bis **L10**, die ungebundenen Textfelder nennen sich **T1** bis **T10**. Die

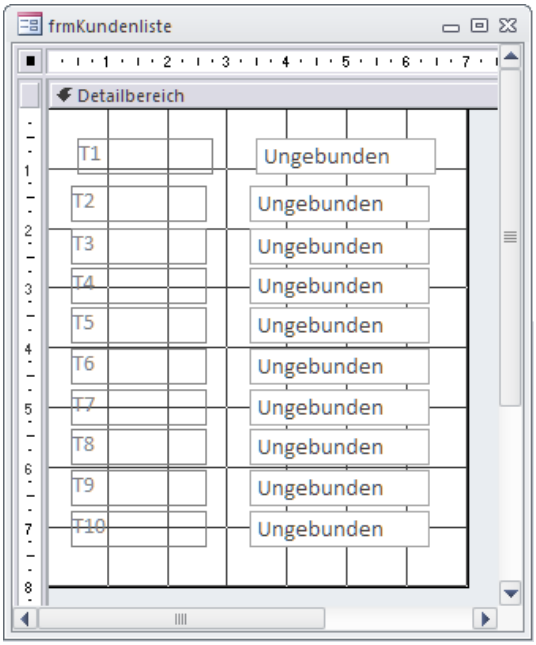


Bild 4: Im Entwurf zeigt das Formular nur zehn Textfelder und Labels.

Aufgabe des Startcodes des Formulars ist es nun, die Labels mit den Feldnamen des ADO-Recordsets zu füllen (**Caption**) und zweitens den Steuerelementinhalt der Textfelder (**ControlSource**) ebenfalls auf diese Namen, damit eine Verknüpfung hergestellt wird. Natürlich hätte man diese Elemente auch alle voreinstellen können, wenn bekannt wäre, wie die **dBase**-Daten aufgebaut sind. Das Formular soll jedoch beliebige **dBase**-Dateien anzeigen können, weshalb dieser allgemeine Aufbau gewählt wurde.

Da die Anzahl der Felder des ADO-Recordsets nicht bekannt ist, müssen die nicht genutzten Spalten zur Laufzeit ausgeblendet werden. Sonst zeigen sich die Textfelder im Datenblatt rechts neben den gültigen Spalten, wie **ID**, **Kundencode**, **Firma**, etwa als **T8**, **T9**, **T10** mit leeren Zellen. Dazu wird die Eigenschaft **ColumnHidden** der entsprechenden Textfelder per Code auf **True** gesetzt. Würden Sie ein Formular in **Formularansicht** verwenden, so müssten Sie lediglich die **Visible**-Eigenschaft der Textfelder und Labels steuern.

Wenn Sie wissen, dass Ihre **dBase**-Dateien auch Tabellen mit mehr als zehn Feldern aufweisen, so können Sie das Formular ja erweitern und mehr Textfelder und Labels einbauen. Ich exportiere etwa meine Bankumsätze aus **StarMoney** in das **dBase**-Format, um sie unter Access für Steuererklärungen analysieren zu können. Die resultierende Datei weist 33 Felder auf. Damit kann das Demoformular in vorliegender Version nicht umgehen.

Zu erwähnen wäre noch, dass die Eigenschaften **Daten eingeben**, **Anfügen zulassen**,

Löschen zulassen und **Bearbeitungen zulassen** des Formulars alle auf **Nein** eingestellt sind. Denn all dies wäre beim ADO-Recordset durchaus möglich, ergibt jedoch wenig Sinn, wenn sich solche Änderungen an den Daten nicht auch in der **dBase**-Datei reflektieren. Lediglich **Filter zulassen** steht auf **Wahr**.

Starten Sie einmal das Formular. Es stellt sich die Ansicht wie in Bild 5 ein, die sich aus dem Laden der Datei **kundenli.dbf** ergibt. Nach fünf Sekunden jedoch kommt es zur Ansicht in Bild 6, was die Datei **orte.dbf** repräsentiert. Das Spiel wiederholt sich alle fünf Sekunden. Das haben wir spaßeshalber nur zur Demonstration dessen eingebaut, dass das Formular dynamisch mit unterschiedlichen Datenquellen umgehen kann.

Listing 4 zeigt den wesentlichen Teil des Formularmoduls. Beim Öffnen wird eine neue Instanz der Klasse **cODBCConnection** in der Variablen **C** erzeugt. **ConnectSilent** weist an, beim Verbinden auf keinen Fall den ODBC-Dialog aufzurufen. Weiter geht es mit **Form_Load**. Dort wird lediglich der Zeitgeber des Formulars auf **5.000 ms** eingestellt, die **Timer**-Routine **Form_Timer** aber auch gleich künstlich aufgerufen. Denn nur die führt

frmKundenliste

KUNDENCODE	FIRMA	STRASSE	ORT	PLZ	LAND	TEL
1014017	Blauer See Delikatessen	Forsterstr. 57	Mannheim	68306	Deutschland	0621-0
1015703	Lazy K Kountry Store	12 Orchestra Terrace	Walla Walla	99362	USA	(509) 5
1022629	Victuailles en stock	2, rue du Commerce	Lyon	69004	Frankreich	78.32.5
1045352	Bottom-Dollar Markets	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Kanada	(604) 5
1045649	Magazzini Alimentari Riuniti	Via Ludovico il Moro 22	Bergamo	24100	Italien	035-64
1053504	FISSA Fabrica Inter. Salchichas S.A.	C/ Moralzarzal, 86	Madrid	28034	Spanien	(91) 55
1056236	Drachenblut Delikatessen	Walserweg 21	Aachen	52066	Deutschland	0241-0
1080714	Richter Supermarkt	Grenzacherweg 237	Genève	1203	Schweiz	0897-0
1097973	Toms Spezialitäten	Luisenstr. 48	Münster	44087	Deutschland	0251-0
1100052	Let's Stop N Shop	87 Polk St.	San Francisco	94117	USA	(415) 5
1103022	LILA-Supermercado	Carrera 52 con Ave. Bolívar #65-98 Llano Largo	Barquisimeto	3508	Venezuela	(9) 331
1106369	La corne d'abondance	67, avenue de l'Europe	Versailles	78000	Frankreich	30.59.8
1160441	Paris spécialités	265, boulevard Charonne	Paris	75012	Frankreich	(1) 42.3
1162821	Pericles Comidas clásicas	Calle Dr. Jorge Cash 321	México D.F.	05033	Mexiko	(5) 552
1186013	Reggiani Caseifici	Strada Provinciale 124	Reggio Emilia	42100	Italien	0522-5
1207561	Rattlesnake Canyon Grocery	2817 Milton Dr.	Albuquerque	87110	USA	(505) 5
1226866	HILARIÓN-Abastos	Carrera 22 con Ave. Carlos Soublette #8-35	San Cristóbal	5022	Venezuela	(5) 555

Datensatz: Kein Filter Suchen

Bild 5: Zur Laufzeit zeigen sich die Daten der **dBase**-Datei korrekt im auf dem **ADO-Recordset** basierenden Formular **frmKundenliste**.

frmKundenliste

ORT	LAND	BREITE	LAENGE	ZONE
Aachen	Deutschland	50 46' N	6 06 O	GMT+1
Aalborg	Dänemark	57 03 N	9 56 O	GMT+1
Aarhus	Dänemark	56 10 N	10 13 O	GMT+1
Aberdeen	Schottland	57 08 N	2 06 W	GMT
Abidjan	Elfenbeinküste	5 19 N	4 02 W	GMT
Abilene	Texas (USA)	32 28 N	99 43 W	GMT-6
Abu Dhabi	Vereinigte Arab Emirate	24 28 N	54 25 O	GMT+4
Acapulco	Mexiko	16 50 N	99 55 W	GMT-6
Accra	Ghana	5 33 N	0 15 W	GMT
Adana	Türkei	37 00 N	35 19 O	GMT+2
Addis Abeba	Äthiopien	9 02 N	38 47 O	GMT+3
Adelaide	Australien	34 56 S	138 36 O	GMT+9
Aden	Jemen	12 46 N	45 01 O	GMT+3
Agadez	Niger	17 00 N	7 56 O	GMT+1
Agadir	Marokko	30 30 N	9 40 W	GMT+1
Ahmedabad	Indien	23 00 N	72 40 O	GMT+5
Akron	Ohio (USA)	41 05 N	81 31 O	GMT-5

Datensatz: 1 von 1234 Kein Filter Suchen

Bild 6: Der **Timer** des Formulars wechselt die Datenherkunft alle 5 Sekunden und führt damit alternativ zur Anzeige der Datei aus **orte.dbf**.

```
Dim rs As ADODB.Recordset
Dim C As cODBCConnection
Dim b As Boolean

Private Sub Form_Open(Cancel As Integer)
  Set C = New cODBCConnection
  C.ConnectSilent = False
End Sub

Private Sub Form_Load()
  DoCmd.Maximize
  Me.TimerInterval = 5000: Form_Timer
End Sub

Private Sub Form_Timer()
  b = Not b
  ChangeDB
End Sub

Private Sub ChangeDB()
  Dim i As Long
  C.ConnectString = "ODBC;" & _
     "Driver={Microsoft dBase Driver (*.dbf)};" & _
     "DriverID=533;Dbq=" & CurrentProject.Path
  C.Connect
  If b Then
    Set rs = C.OpenRecordsetADO( _
               "SELECT * FROM KUNDENLI.DBF")
  Else
    Set rs = C.OpenRecordsetADO( _
               "SELECT * FROM ORTE.DBF")
  End If
  For i = 1 To 10
    With Me.Controls("T" & CStr(i))
        If i > rs.Fields.Count Then
          Me.Controls("T" & CStr(i)).ColumnHidden = True
        Else
          .ColumnHidden = False
          .ControlSource = rs.Fields(i - 1).Name
          Me.Controls("L" & CStr(i)).Caption = _
            .ControlSource
        End If
    End With
  Next i
  Set Me.Recordset = rs
End Sub
```

Listing 4: Das fast komplette Modul des Formulars **frmKundenliste**

zur Anzeige der **dBase**-Daten. Sie verneint die modulweit gültige Boolean-Variable **b**, so dass diese bei jedem Zeitgeberaufruf von **True** auf **False** und umgekehrt wechselt. Erst die nun aufgerufene zentrale Prozedur **ChangeDB** wertet die Variable aus.

Sie bereitet zuerst den **ConnectString** auf und gibt in diesem den **ODBC-Treiber** und den Ort der **dBase**-Dateien an. Die **ODBCDirect**-Verbindung wird dann mit **Connect** aufgebaut. Je nach Zustand der Variablen **b** verzweigt die Routine in Zeilen, die über **SELECT**-Statements **ADO-Recordsets** auf verschiedene **dbf**-Dateien öffnen.

Nun können die Felder der Recordsets inspiziert werden. In der Schleife von **1** bis **10** werden alle Controls des Formulars durchlaufen. Ist **i** größer, als die Anzahl der Felder (**Fields.Count**), so wird **ColumnHidden** auf **True** gesetzt. Andernfalls werden der Steuerelementinhalt der Textfelder (**ControlSource**) und die Beschreibung der Labels (**Caption**) auf den Feldnamen der Recordsets eingestellt (**Fields(i-1).Name**). Die Steuerelemente werden dabei jeweils über ein **L** oder ein **T** plus Zähler-String angesprochen. Abschließend wird das Formular-Recordset per **Set** auf das ADO-Recordset **rs** gesetzt.

Access stellt nun die Verbindung zwischen Steuerelementen und Datenherkunft automatisch ein. Die Datenblattansicht zeigt die **dBase**-Daten jetzt an. Im Prinzip ließe sich das Klassenmodul auch noch so erweitern, dass sich Änderungen am ADO-Recordset etwa über eine Methode **SaveADO** über **ODBCDirect** zurück auf die Server-Tabellen übertragen. Das allerdings wäre eine heikle Angelegenheit, da die Änderungen möglicherweise nicht bedingungskonform wären, weil die Tabelle auf dem Server etwa **Constraints** aufweist oder Fremdschlüssel, die verletzt werden könnten.

Die ganze Angelegenheit können Sie auch für andere **DBMS** einsetzen. Geben Sie im **ConnectString** etwa einfach einen **MySQL**-Treiber an und verbinden sich zu Tabellen, **Views** oder **SPs** auf einem **MySQL**-Server.

Das SQL Server Management Studio

Wenn Sie mit dem Microsoft SQL Server arbeiten und SQL Server-Instanzen, die enthaltenen Datenbanken mit Tabellen und weiteren Objekten, Sicherheitselemente wie Anmeldungen, Benutzer oder Benutzergruppen und mehr verwalten wollen, kommen Sie um das SQL Server Management Studio nicht herum. Sie können zwar theoretisch alle Aktionen, die Sie mit diesem Tool durchführen, auch mit T-SQL direkt aufrufen. Aber dafür müssen Sie schon profunde Kenntnisse dieser Sprache in Zusammenhang mit der Verwaltung einer SQL Server-Instanz haben. Dieser Beitrag liefert die Grundlagen für den Umgang mit dem Administrationstool für den SQL Server.

Das SQL Server Management Studio können Sie in der aktuellsten Version nutzen, auch wenn Sie nicht die aktuellste SQL Server-Installation auf Ihrem Rechner haben. Die aktuelle Version finden Sie recht schnell, wenn Sie bei Google nach den Begriffen **sql server management studio 2017 download** suchen. In diesem Beitrag arbeiten wir mit der Version 2017 dieses Tools.

Genau genommen können Sie das SQL Server Management Studio auch komplett ohne SQL Server auf einem lokalen Rechner nutzen – Sie können sich damit auch etwa mit einer Datenbank im Internet verbinden, also etwa einer Azure-Datenbank.

Voraussetzung für die folgenden Schritte ist natürlich, dass bereits eine SQL Server Instanz läuft, die von diesem Rechner aus erreichbar ist – es reicht auch eine LocalDb (siehe Beitrag **Access und LocalDb**, **www.access-im-unternehmen.de/1057**). Wenn Sie beispielsweise eine aktuelle Version von Visual Studio installiert haben, sollte sich bereits eine LocalDb-Installation auf Ihrem Rechner befinden.

Wenn Sie beispielsweise LocalDb verwenden wollen, geben Sie im nun erscheinenden Dialog **Verbindung mit SQL Server herstellen** unter **Servername** den Wert

Nach dem Start erscheint gleich das Fenster aus Bild 1. Da Sie zu diesem Zeitpunkt noch nicht mit einem SQL Server verbunden sind, holen Sie das gleich nach – am schnellsten durch einen Klick auf die Schaltfläche **Verbinden**. Dies öffnet ein Menü, in dem für die Verbindung mit einem SQL Server, gleich ob lokal oder über das Internet erreichbar, der Eintrag **Datenbankmodul...** interessant ist.

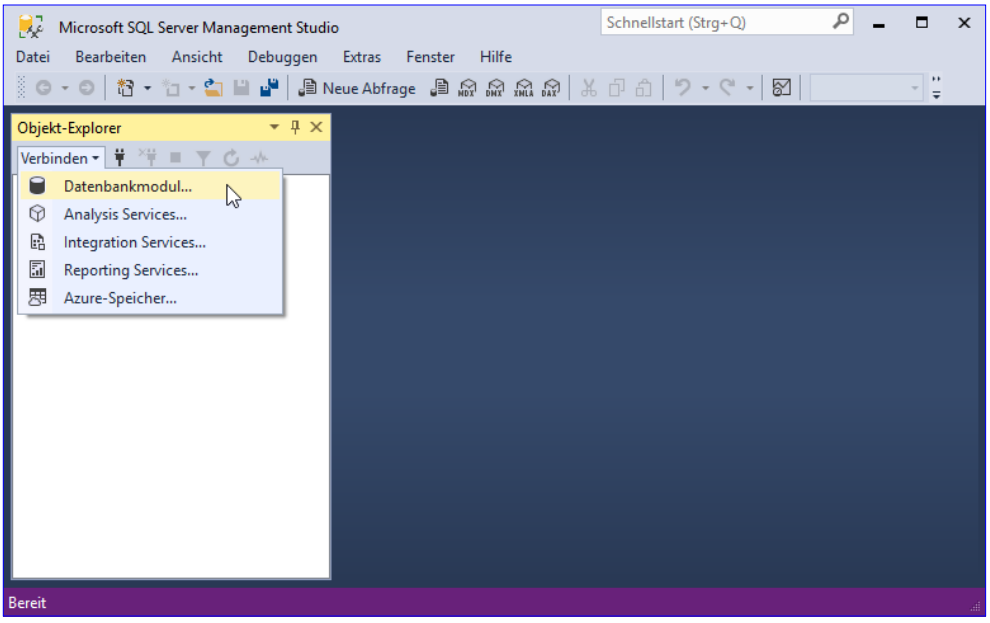


Bild 1: Das SQL Server Management Studio nach dem Öffnen

(localdb)\SQLServerLocalDb ein (siehe Bild 2). Danach erscheinen im Objekt-Explorer ein Eintrag für den Server sowie die Verzeichnisse mit den Elementen des SQL Servers. Wenn Sie neben LocalDb vielleicht noch eine Vollversion des SQL Servers verwalten wollen, erstellen Sie über die **Verbinden**-Schaltfläche einfach noch eine zweite Verbindung. Die Verbindungen werden dann einfach untereinander im Objekt-Explorer angezeigt.

Sie können auch die gleiche SQL Server-Instanz mehrfach verbinden, um beispielsweise den Zugriff unter verschiedenen Benutzern zu testen. Das gelingt allerdings nur bei der SQL Server-Authentifizierung – für die Anmeldung unter einem anderen Windows-Benutzer benötigen Sie neue Instanz des SQL Server Management Studios. Wie Sie das machen, erfahren Sie im Beitrag **Authentifizierung im SQL Server testen** (**www.access-im-unternehmen.de/1157**).

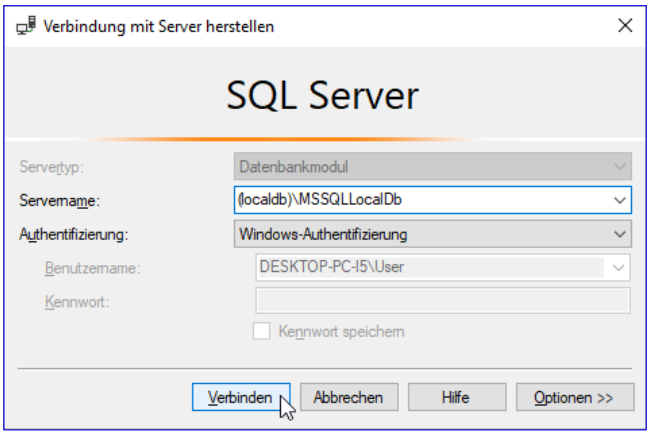


Bild 2: Anmelden an LocalDb

Elemente einer SQL Server-Verbindung

Der Objekt-Explorer zeigt nun den Eintrag für die Verbindung an sowie einige Unterordner:

- **Datenbanken**: Enthält die Datenbanken der verbundenen SQL Server-Instanz sowie deren Elemente wie Tabellen, gespeicherte Prozeduren, Sicherheitsinformationen und mehr
- **Sicherheit**: Enthält die Elemente zum Verwalten von Benutzern und Berechtigungen
- **Serverobjekte**: Enthält Informationen über Sicherheitsmedien, Verbindungsserver und Trigger
- **Replikation**: Stellt Replikationsfunktionen bereit.
- **Verwaltung**: Enthält unter anderem die Einträge **Erweiterte Ereignisse** und **SQL Server-Protokolle**. Diese Funktion erläutern wir im Beitrag **SQL Server: Zugriffe untersuchen mit XEvents** (**www.access-im-unternehmen.de/1158**) detailliert. Die SQL Server-Protokolle schauen wir uns gleich an.

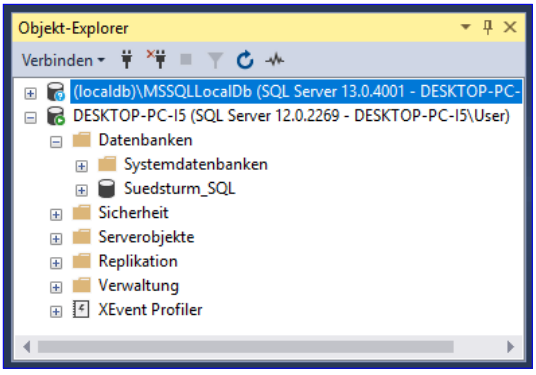


Bild 3: SQL Server mit Datenbanken

Datenbanken verwalten

Wenn Sie das Verzeichnis **Datenbanken** aufklappen, finden Sie die auf dem Server befindlichen Datenbanken vor (siehe Bild 3). Wenn Sie noch keine Datenbank haben, können Sie die Beispieldatenbank aus dem Download dieses Beitrags installieren. Wie das gelingt, lesen Sie im Beitrag **SQL Server-Datenbank kopieren** (**www.access-im-unternehmen.de/1153**).

Wie Sie eigene Datenbanken erstellen, zeigen wir später in einem eigenen Beitrag.

Sicherheit

Den Bereich **Sicherheit** mit den Anmeldungen für Windows-Benutzer und -Benutzergruppen sowie für SQL Server-Benutzer besprechen wir ausführlich in den Beiträgen **SQL Server: Sicherheit und Benutzerverwaltung** (**www.access-im-unternehmen.de/1154**) und

SQL Server-Authentifizierung (**www.access-im-unternehmen.de/1155**).

SQL Server-Protokolle

Unter dem Eintrag **Verwaltung|SQL Server-Protokolle** finden Sie verschiedene Einträge, deren Namen beispielsweise mit **Aktuell...** oder **Archiv...** beginnen (siehe Bild 4).

Eine solche Datei öffnen Sie per Doppelklick auf den jeweiligen Eintrag. In den Protokollen notiert der SQL Server Meldungen über den Start des SQL Server-Dienstes, Ausführungen von Sicherungen oder Starts von Datenbanken. Die Details im unteren Bereich liefern meist keine besonderen Informationen (siehe Bild 5).

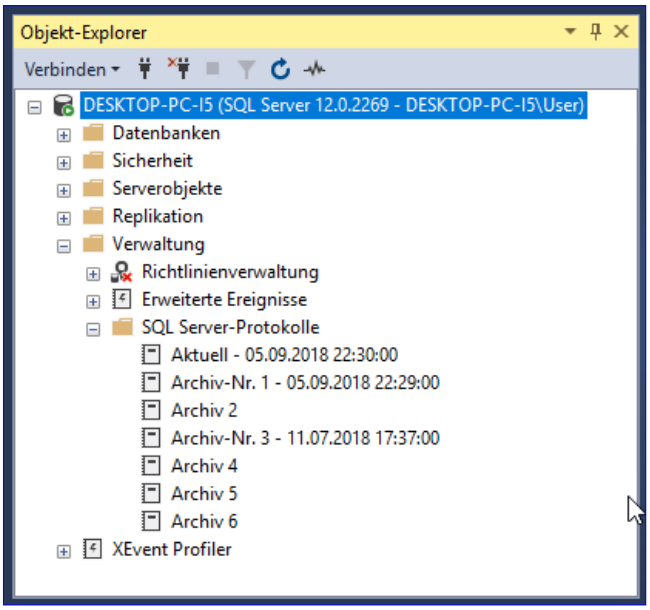


Bild 4: SQL Server-Protokolle

SQL Server Agent

Der SQL Server Agent, der übrigens nicht in der LocalDb oder der Express-Edition des SQL Servers enthalten ist, ist eine Art Taskplaner für den SQL Server. Mit ihm können Sie Aufgaben definieren, die zu fest vorgegebenen Zeiten ausgeführt werden. Wir werden dieses Tool allerdings bei

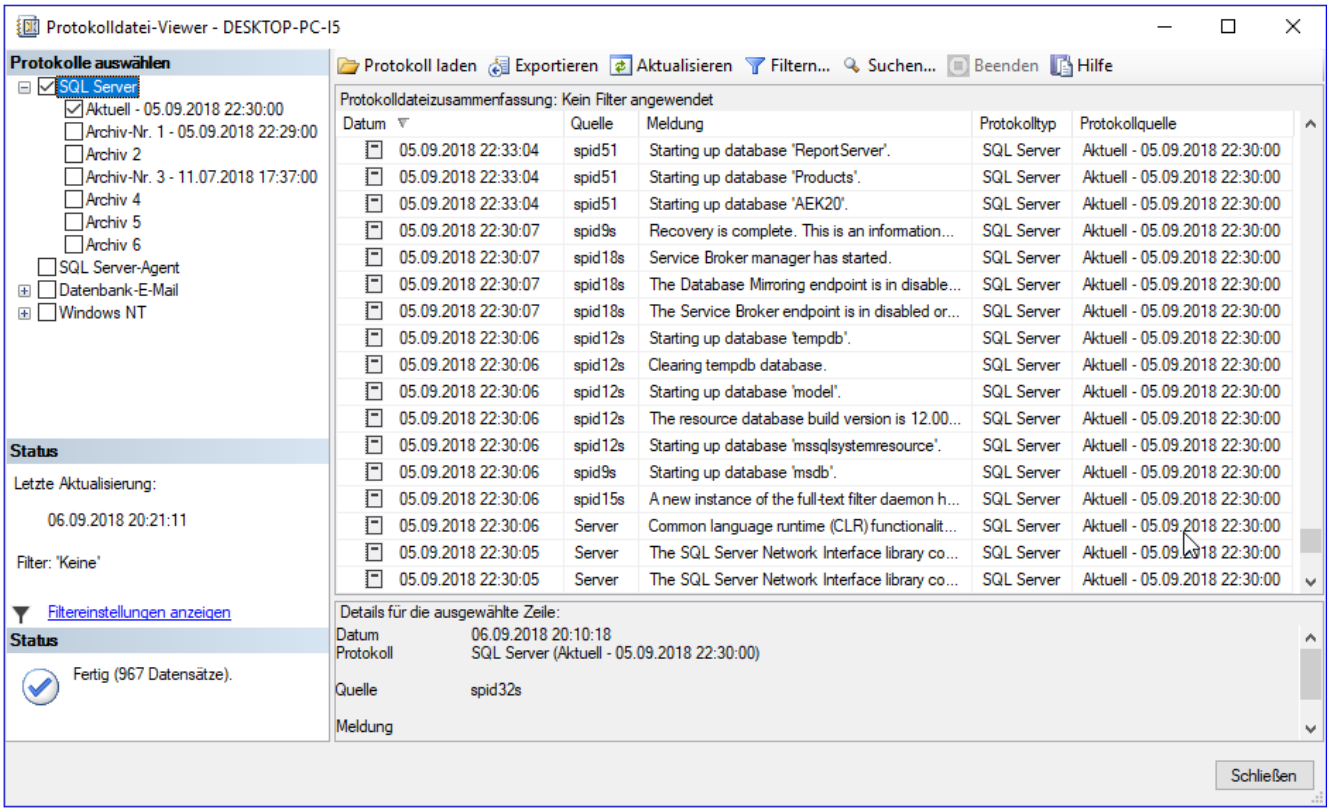


Bild 5: Ein SQL Server-Protokoll mit den aktuellsten Informationen

der Arbeit mit dem SQL Server als Backend von Access-Anwendungen zunächst nicht nutzen, außerdem wollen wir uns auf die Features der frei verfügbaren Versionen konzentrieren.

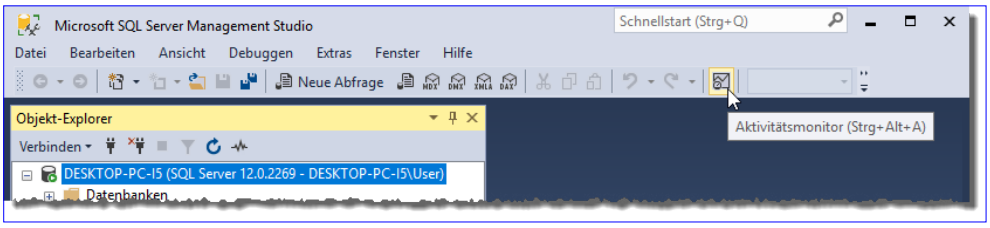


Bild 6: Starten des Aktivitätsmonitors

Der Aktivitätsmonitor

Der Aktivitätsmonitor liefert Informationen über die Auslastung des SQL Servers. In den Beispielen der Beiträge zum SQL Server werden wir diesen vermutlich nicht ausreizen, allerdings wollen wir dieses Tool dennoch kurz vorstellen. Sie starten den Aktivitätsmonitor über die unscheinbare Schaltfläche in der Symbolleiste des SQL Server Management Studios (siehe Bild 6).

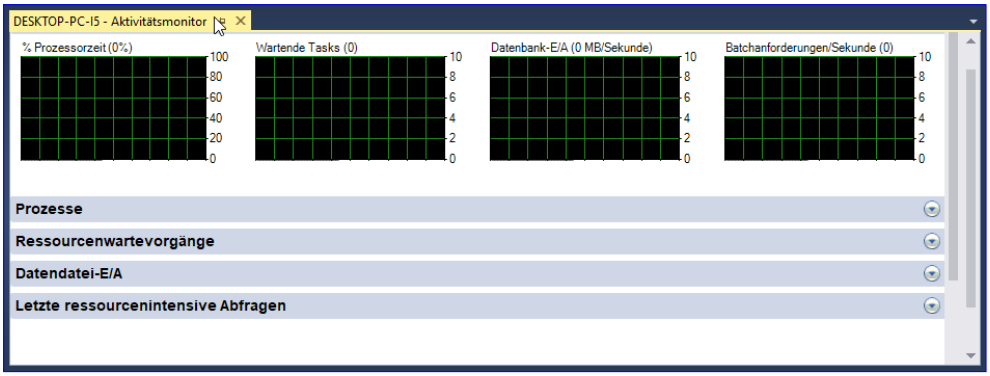


Bild 7: Die Diagramme des Aktivitätsmonitors

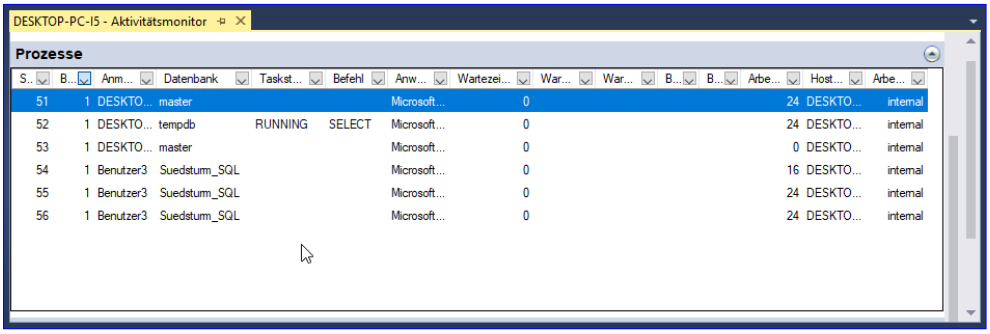


Bild 8: Die aktuellen Prozesse der SQL Server-Instanz

Im oberen Bereich finden Sie vier Diagramme für die folgenden Informationen (siehe Bild 7). Hier tut sich während der Arbeit an diesem Beitrag erwartungsgemäß nicht allzu viel:

- Prozessorzeit
- Wartende Tasks
- Datenbank-E/A
- Batchanforderungen/Sekunde

Darunter gibt es vier Bereiche für die folgenden Informationen, die Sie nach Wunsch aufklappen können:

- Prozesse
- Ressourcenwartevorgänge
- Dateidatei-E/A
- Letzte resourcenintensive Abfragen

Interessant ist hier zum Beispiel die Ansicht der Prozesse (siehe Bild 8). Hier finden Sie wichtige Informationen über jeden Prozess wie etwa die Datenbank, auf die zugegriffen wird, welcher Benutzer den Zugriff ausführt, von welcher Anwendung dieser erfolgt und von welchem Client.

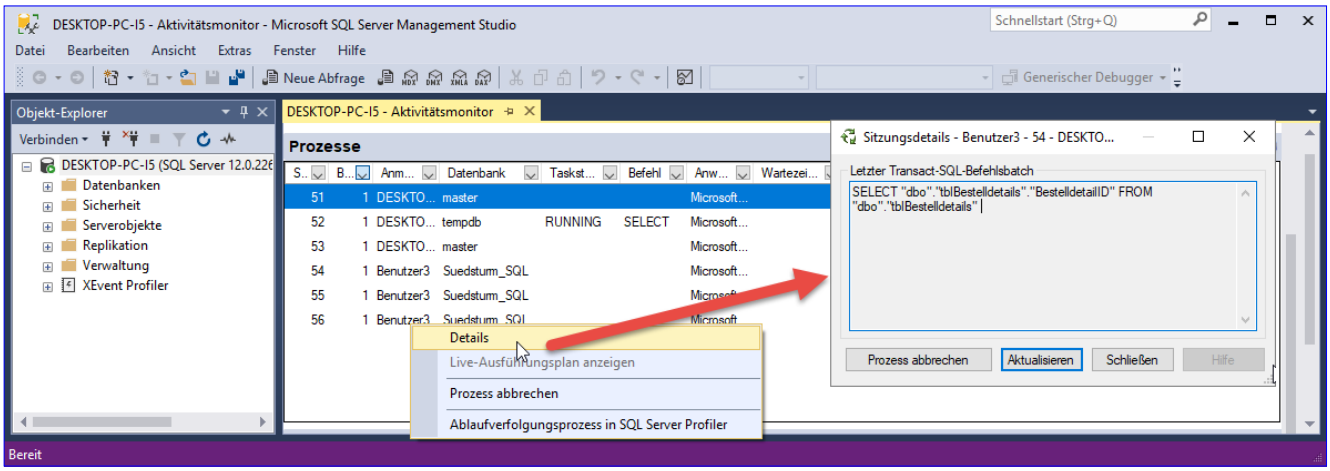


Bild 9: Per Kontextmenü rufen Sie die Details auf, etwa den Text der ausgeführten Abfrage

Wenn Sie mit der rechten Maustaste auf einen der Einträge in der Liste der Prozesse klicken und den Befehl **Details** ausführen, erhalten Sie sogar den SQL-Code der soeben aufgerufenen Anweisung (siehe Bild 9).

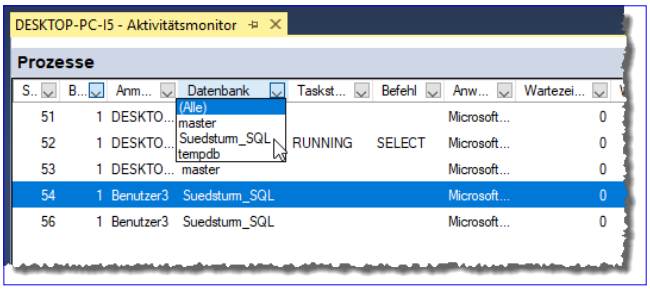


Bild 10: Aufrufen eines der Spaltenfilter

Wenn Ihnen dieser Dialog zu unübersichtlich wird, können Sie jede Spalte nach den enthaltenen Werten filtern. Dazu klicken Sie einfach für die jeweilige Spalte auf die Tasten mit dem Pfeil nach unten (siehe Bild 10).

Interessant ist auch der Bereich **Letzte ressourcenintensiven Abfragen** (siehe Bild 11). Hier finden Sie die zuletzt ausgeführten Abfragen mit einer nicht optimalen Performance. In der aktuellen Ansicht finden Sie allerdings noch die Abfragen aller Datenbanken. Sie können hier auch einen Filter auf eine der Datenbanken setzen und so beispielsweise nur die Abfragen anzeigen, die von Ihrem aktuell in der Programmierung befindlichen Access-Frontend aufgerufen werden.

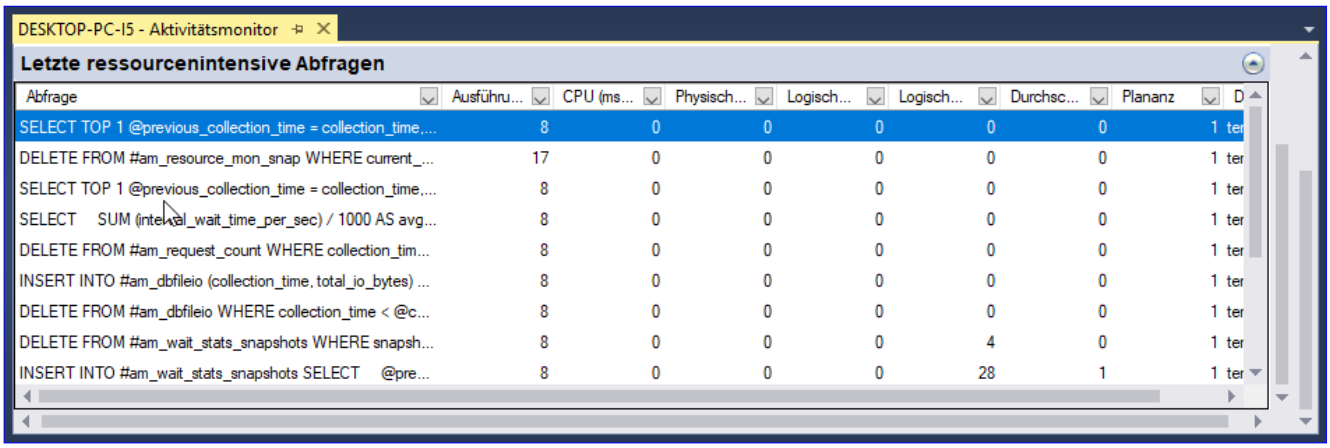


Bild 11: Per Kontextmenü rufen Sie die Details auf, etwa den Text der ausgeführten Abfrage

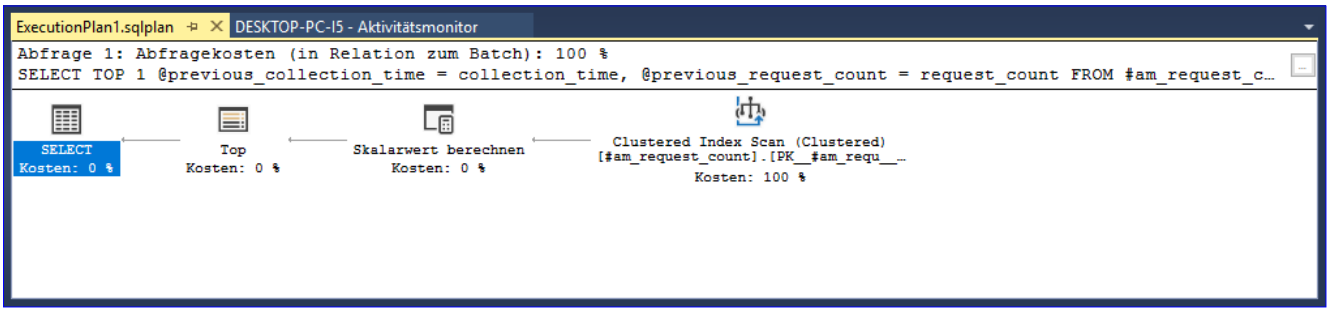


Bild 12: Aufrufen eines der Spaltenfilter

Wenn Sie mit der rechten Maustaste auf einen der Einträge klicken, erscheint ebenfalls ein Kontextmenü. Dieses zeigt zum Beispiel den Befehl **Ausführungsplan anzeigen** an, der das Fenster aus Bild 12 öffnet und den Ausführungsplan für die jeweilige Abfrage anzeigt. Der andere Befehl des Kontextmenüs namens **Abfragetext bearbeiten** liefert den SQL-Code der Abfrage.

Berichte

Nein, hier geht es nicht um die Objekte von Access zur formatierten Ausgabe der Daten einer Datenbank. Unter SQL Server finden Sie im Kontextmenü fast aller Objekte einen Eintrag namens **Berichte**, der eine Reihe von Berichten anzeigt, die Sie für das jeweilige Objekt ausgeben können. Für manche Objekte gibt es eine ganze Reihe von Berichten, andere – wie etwa die Tabellen einer Datenbank – bieten lediglich den Eintrag **Benutzerdefinierte Berichte...**, der einen Dateidialog zum Öffnen eines benutzerdefinierten Berichts anzeigt.

Der Eintrag des jeweiligen Servers jedoch liefert, wie in zu Bild 13 erkennen, einige Standardberichte. Wir haben im Screenshot einige Berichte hervorgehoben, die interessante Informationen über die SQL Server-Instanz liefern:

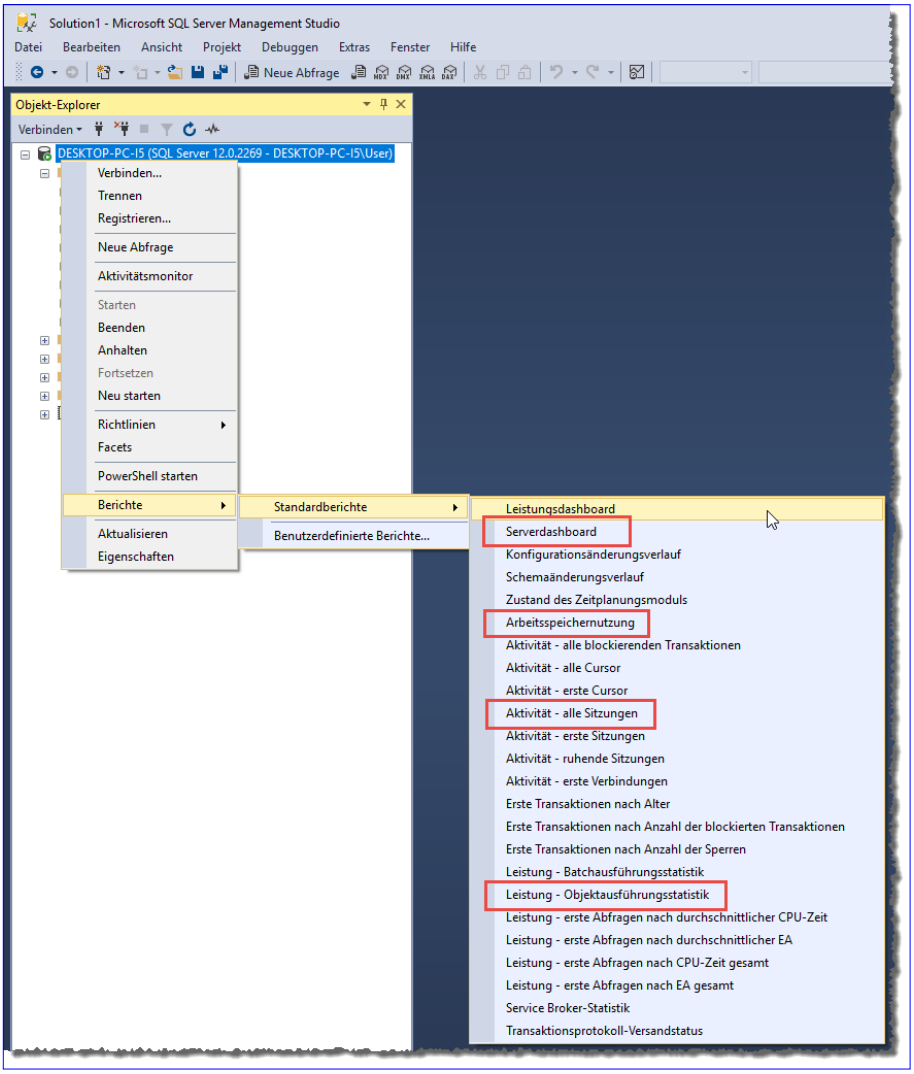


Bild 13: Auswahl der Berichte für die SQL Server-Instanz

SQL Server-Datenbanken kopieren

Für die Beispiele der Beiträge in diesem Heft und in weiteren Ausgaben wollen wir mit Beispieldatenbanken arbeiten, die wir nicht immer komplett neu erstellen, sondern aus einer von mehreren Quellen auf dem Rechner, auf dem Sie die Beispiele ausprobieren, wiederherstellen wollen. Dazu gibt es drei gängige Methoden, die wir in diesem Artikel vorstellen wollen: den Assistenten zum Kopieren von Datenbanken, das Wiederherstellen einer Datenbanksicherung sowie das Wiederherstellen mithilfe eines Skripts. Wir wollen einen Blick auf die letzteren beiden Varianten werfen.

Voraussetzungen

Wenn Sie unsere Beispiele zur Nutzung von Access mit dem SQL Servers als Datenbank-Backend verwenden wollen, benötigen Sie eine Instanz des SQL Servers auf Ihrem System sowie die entsprechende Beispieldatenbank. Die Instanz können Sie sich beispielsweise in Form von LocalDb holen oder als SQL Server Express. Die Verwendung von LocalDb beschreiben wir im Artikel **Access und LocalDb** (**http://www.access-im-unternehmen.de/1057**). Außerdem arbeiten wir mit dem SQL Server Management Studio, um den SQL Server und die darin enthaltenen Datenbanken zu verwalten.

Kopieren von Access-Daten

In der Access-Welt war das Kopieren von Daten überhaupt kein Problem. Sie haben einfach die komplette Datenbankanwendung kopiert und auf dem Zielrechner wieder eingefügt. Sofern dort Access installiert ist, können Sie direkt loslegen. Wenn Sie die Daten hingegen in einem Backend speichern, sind neben dem Kopieren des Backends noch ein paar weitere Schritte nötig, um etwa das Frontend erneut mit dem Backend zu verknüpfen. Mit dem SQL Server ist das alles ein wenig komplizierter, denn Sie können nicht einfach eine Datei von A nach B kopieren und dann direkt wieder auf die Daten zugreifen. Stattdessen müssen wir die Datenbank erst auf Rechner A in einer bestimmten Art und Weise sichern und auf Rechner B wiederherstellen.

Für unseren Fall, indem wir nur einfache Beispieldatenbanken aus dem Download des Magazins auf Ihrem Rechner installieren wollen, damit Sie die Beispiele aus den Beiträgen damit ausprobieren können, gibt es zwei einfache Strategien. Die erste ist das Erstellen einer Sicherung auf dem ersten Rechner und das Wiederherstellen aus der Sicherungsdatei auf dem Zielrechner. Die zweite ist, auf dem Quellrechner einfach ein Skript generieren zu lassen, dass die Definition der Tabellen, Felder und Restriktionen sowie die enthaltenen Daten enthält. Dieses können Sie dann auf dem Zielrechner einfach wieder ausführen, um die Datenbank neu zu erstellen.

Sichern und wiederherstellen

Als Erstes schauen wir uns die Variante über die Sicherungs- und Wiederherstellungsfunktion von SQL Server an. Hier öffnen Sie das SQL Server Management Studio und wählen für die zu sichernde Datenbank den Kontextmenü-Eintrag **Tasks|Sichern...** aus (siehe Bild 1).

Es erscheint der Dialog **Datenbank sichern - <Datenbankname>**, der wie in Bild 2 aussieht. Hier legen Sie einige wichtige Informationen fest:

- Unter **Datenbank** erscheint die Datenbank, für die Sie den Dialog aufgerufen haben.
- **Sicherungstyp** gibt an, ob Sie eine vollständige Sicherung der Datenbank durchführen wollen oder eine differenzielle Sicherung. Letzteres ist für uns aktuell nicht interessant, weil dies für die Sicherung im laufenden Betrieb und die Wiederherstellung nach einem

Datenverlust gedacht ist. Wir wollen die Datenbank ja nur von A nach B kopieren. Behalten Sie also den Wert **Vollständig** bei.

- **Sichern nach:** Hier geben Sie an, ob auf einen Datenträger oder zu einer URL gesichert werden soll. Wir wählen **Datenträger**.

Darunter geben Sie noch an, in welche Datei die Sicherung gespeichert werden soll. SQL Server legt automatisch eine Datei fest, die den Namen der Datenbank sowie die Dateiendung **.bak** trägt. Die Datei landet im Verzeichnis Backup des SQL Server-Verzeichnisses. Sie können hier einen beliebigen anderen Ordner oder auch einen anderen Dateinamen angeben. Dazu müssen Sie allerdings den vorhandenen Sicherungsort mit der **Entfernen**-Schaltfläche löschen und mit **Hinzufügen...** einen neuen Speicherort anlegen. Beim Hinzufügen stellt der SQL Server aber immerhin den zuletzt verwendeten Ordner im Dialog zum Festlegen eines neuen Sicherungsziels ein (siehe Bild 3).

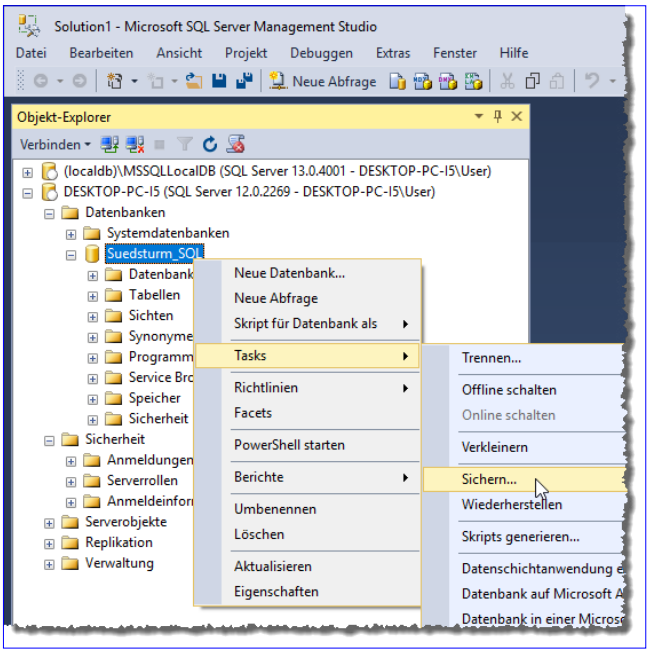


Bild 1: Aufrufen des Dialogs zum Sichern einer Datenbank

Schließlich aktivieren Sie noch die Option **Kopiesicherung**. Auf diesem Wege stellen Sie sicher, dass eventuell vorhandene geplante Vollsicherungen, differenzielle Sicherungen und Transaktionssicherungen nicht beeinflusst werden.

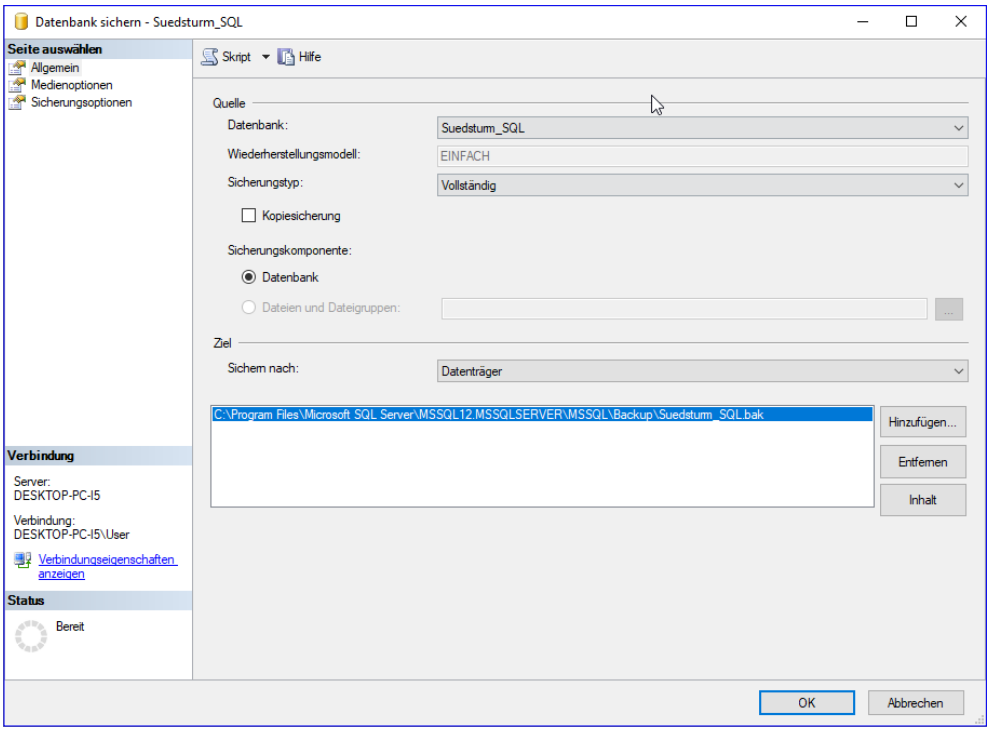


Bild 2: Dialogs zum Sichern einer Datenbank

Die übrigen Einstellungen behalten wir bei, da wir ja nur eine Sicherung für die Weitergabe erstellen wollen und keine, die einen Ausfall des Systems mit Datenverlust kompensieren soll. Klicken Sie nun auf **OK**, wird die Sicherung direkt ausgeführt. Der erfolgreiche Abschluss der Sicherung wird in der Meldung aus Bild 4 bestätigt.

Damit können wir uns nun der Wiederherstellung der

Datenbank etwa auf einem anderen Rechner oder auch einer anderen SQL Server-Instanz widmen. Ich habe an dieser Stelle übrigens festgestellt, dass sich die Sicherung nicht in ein Verzeichnis des Benutzerkontos speichern ließ – mal sehen, ob sich daran etwas ändern lässt.

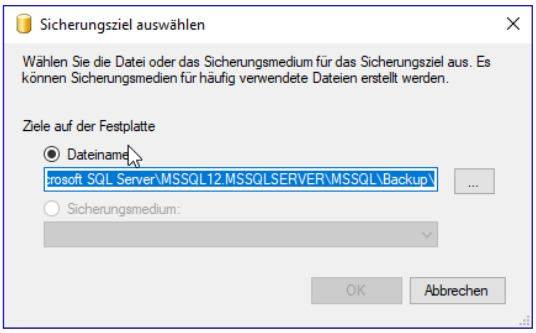


Bild 3: Anlegen eines neuen Ziels für die Sicherung

Diesen Befehl geben Sie im SQL Server Management Studio in eine neue Abfrage ein, die Sie über den Kontextmenübefehl **Neue Abfrage** des Eintrags für die jeweilige Datenbank im Objekt-Explorer anlegen. Führen Sie die Abfrage dann aus, erhalten Sie das Ergebnis aus Bild 5.

Vollsicherung per T-SQL

Annähernd alle Aufgaben, die Sie über die Benutzeroberfläche des SQL Server Management Studios erledigen, können Sie auch per T-SQL als Befehl oder Folge von Befehlen ausführen. In diesem Fall handelt es sich um den Befehl **BACKUP DATABASE**, der für unsere Konfiguration wie folgt aussieht:

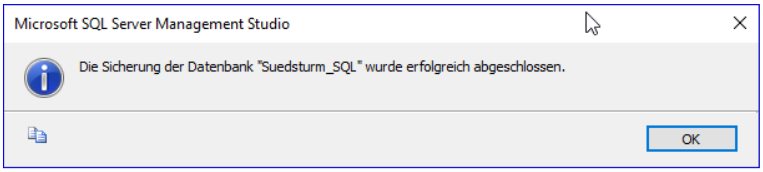


Bild 4: Erfolgreicher Abschluss einer Sicherung

```
BACKUP DATABASE [Suedsturm_SQL]
TO DISK = N'C:\Program Files\Microsoft SQL Server\MSSQL12.
MSSQLSERVER\MSSQL\Backup\Suedsturm_SQL.bak'
WITH NOFORMAT, NOINIT, NAME = N'Suedsturm_SQL-Vollständig
Datenbank Sichern', SKIP,
NOREWIND, NOUNLOAD, STATS = 10
```

An dieser Stelle haben wir nun probiert, den Namen der Backupdatei so zu ändern, dass das Backup in ein Unterverzeichnis des Benutzerverzeichnisses kopiert wird, aber auch hier gelingt dies nicht – wir erhalten Fehler 5, **Zugriff verweigert**. Wir müssen uns also damit begnügen, dass die Sicherungsdatei an einem anderen Ort gespeichert werden muss.

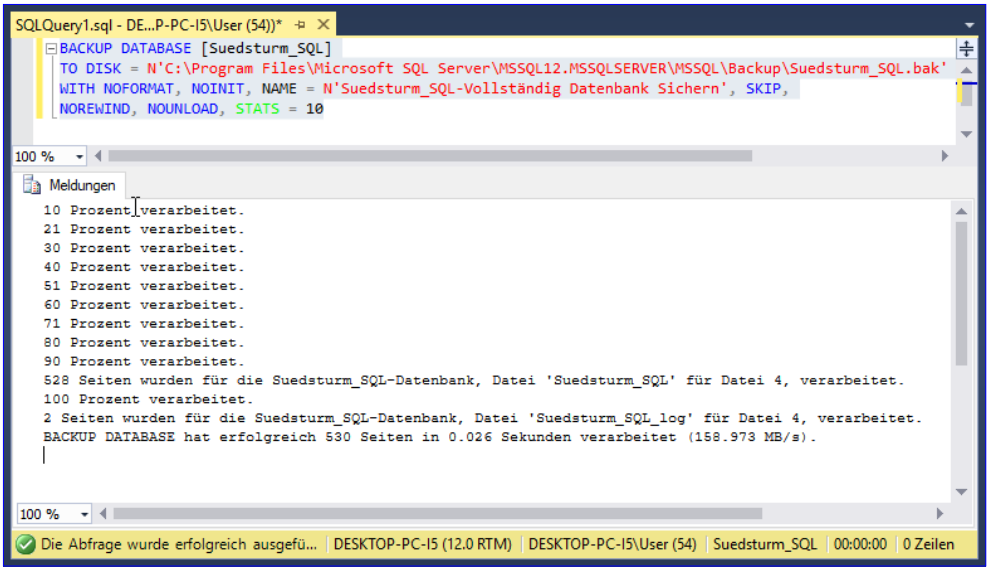


Bild 5: Sicherung per T-SQL

Wiederherstellen aus einer Sicherungsdatei

Damit kommen wir zum zweiten Teil des Kopierens einer Datenbank – der Wiederherstellung. Diese beginnt wieder im Objekt-Explorer des SQL Server Management Studios. Hier wählen Sie für den Eintrag Datenbanken den Kontextmenü-Eintrag **Datenbank wiederherstellen...** aus (siehe Bild 6).

Dies öffnet den Dialog aus Bild 7. Hier wählen wir unter **Quelle** zunächst die

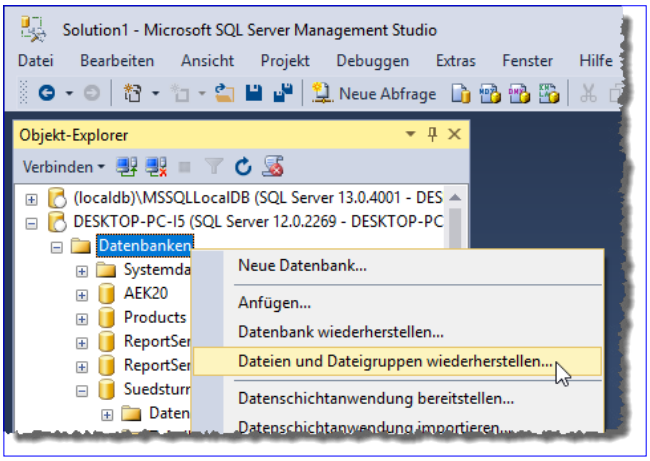


Bild 6: Start der Wiederherstellung einer Datenbank

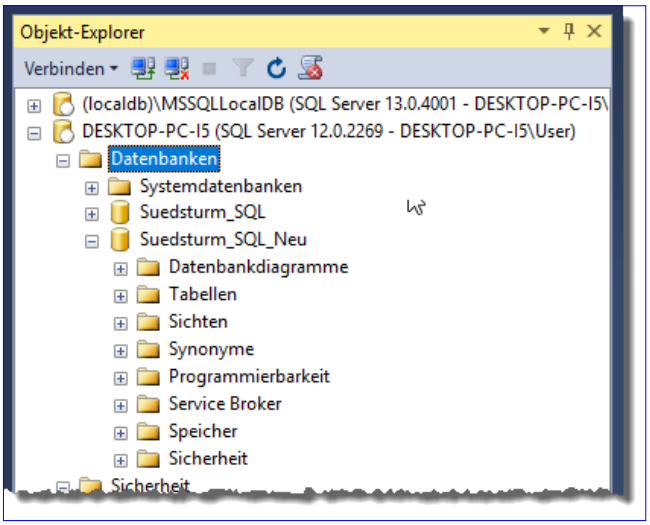


Bild 8: Die neue Datenbank im Objekt-Explorer

Option **Medium** aus. Damit können wir dann über die Schaltfläche mit den drei Punkten die Datei auswählen, aus der wir die Sicherung wiederherstellen wollen. Sobald dies geschehen ist, werden die meisten Felder automatisch mit den Informationen aus der Sicherung gefüllt. Da wir uns noch in der gleichen SQL Server-Instanz befinden und nicht die vorhandene Datenbank überschreiben wollen, geben Sie unter **Ziel** einen anderen Namen für **Datenbank:** an, hier **Suedsturm_SQL_Neu**. Mit einem Klick auf die Schaltfläche **OK** starten wir die Wiederherstellung, die kurz danach durch eine entsprechende Meldung als erfolgreich bestätigt wird.

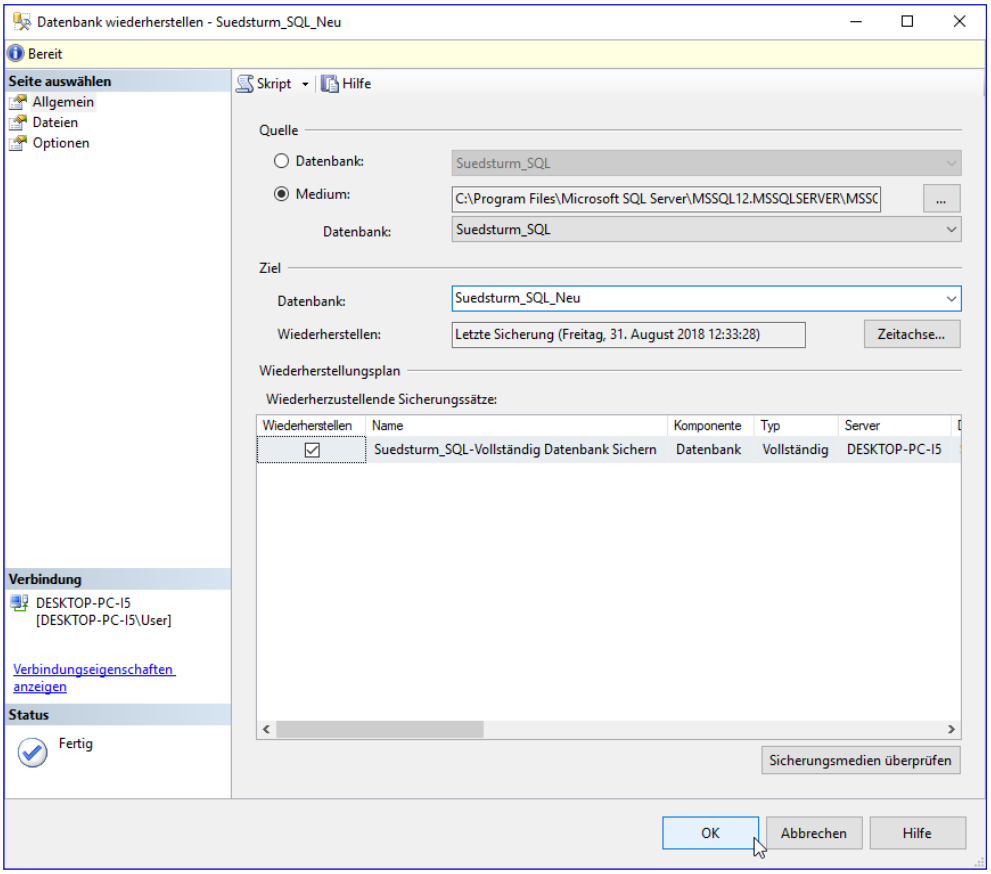


Bild 7: Wiederherstellen einer Sicherung

Anschließend finden wir wie in Bild 8 die neue Datenbank unter dem angegebenen Namen im Objekt-Explorer vor.

Auch diesen Schritt können Sie ganz einfach mit einer T-SQL-Anweisung realisieren. Der dazu benötigte Befehl lautet diesmal:

SQL Server: Sicherheit und Benutzerverwaltung

Der Wechsel von einem Access-Backend zum SQL Server als Backend-System hat – neben der Performance und dem Speicherplatz – oft einen entscheidenden Grund: Access schützt die Daten nicht ausreichend vor fremden Zugriffen. Hier hat der SQL Server deutlich die Nase vorn, denn er bietet verschiedene Möglichkeiten der Sicherung der Daten vor dem unbefugten Zugriff. Dieser Artikel zeigt, wie das Sicherheitssystem und die Benutzerverwaltung im SQL Server funktionieren.

Mit Access 2010 hat Microsoft sich endgültig vom Sicherheitssystem verabschiedet. Zuvor gab es unter Access noch die sogenannte Arbeitsgruppendatei (**.mdw**) und einige Elemente in der Benutzeroberfläche, mit denen Sie Benutzergruppen, Benutzer und deren Zugriffsrechte auf die einzelnen Elemente einer Datenbank beeinflussen konnten.

War einer Access-Datenbank eine Arbeitsgruppendatei zugeordnet, fragte Access beim Öffnen der Anwendung den Namen und das Kennwort des Benutzers ab und entschied dann, ob dieser die Anwendung überhaupt öffnen durfte. Anschließend gab es Freigaben zum Öffnen der verschiedenen Access-Objekte nur entsprechend der Zugriffsberechtigungen des Benutzers oder der Benutzergruppe, der dieser Benutzer zugeordnet war.

Allerdings war dieses System schon lange nicht mehr sicher und bot nur einen mittelmäßigen Schutz vor Datendiebstahl. Mittelmäßiges Interesse am Knacken der Datenbank vorausgesetzt, war es leicht möglich, an die enthaltenen Daten zu gelangen. Wohl auch deshalb hat sich Microsoft entschlossen, dieses Sicherheitssystem vor einigen Jahren abzuschaffen und die Access-Entwickler und auch die Benutzer nicht mehr in scheinbarer Sicherheit zu wiegen.

Wer überhaupt ein Sicherheitssystem verwenden wollte, musste umsteigen oder sich auf das allzu rudimentäre Datenbankkennwort verlassen – das aber dann entweder den Zugriff komplett verwehrte oder aber den Datenbankinhalt vollständig freigegeben hat.

Sicherheit im SQL Server

Im SQL Server gibt es erheblich mehr Features, um die in den Tabellen enthaltenen Daten vor neugierigen Blicken oder auch vor unbefugtem Verändern zu schützen. Die Anmeldung an einer Datenbank erfolgt dabei auf zwei verschiedene Arten:

- Authentifizierung über eine Windows-Anmeldung
- Authentifizierung über ein SQL Server-Konto

Authentifizierung über eine Windows-Anmeldung

Die erste Variante ist sinnvoll, wenn Sie über eine Domäne verfügen, also einen Bereich im Netzwerk, in dem die Benutzerrechte für verschiedene Benutzer zentral verwaltet werden. Hier erhält jeder Benutzer bereits in der Domäne ein eigenes Benutzerkonto, mit der sich auf seinem Rechner anmelden und je nach Berechtigungen auf die Daten und Geräte im Netzwerk zugreifen kann.

Dazu gehören dann auch die SQL Server, die in der Domäne erreichbar sind und die enthaltenen Datenbanken. Im SQL Server werden dann Konten für die in der Domäne bereits vorhandenen Benutzer oder Benutzergruppen angelegt.

Meldet sich der Benutzer dann unter seinem Windows-Konto am Rechner und somit in der Domäne an und versucht dann, auf den SQL Server zuzugreifen, werden seine Windows-Anmeldedaten daraufhin geprüft, ob das Konto auch im SQL Server vorliegt. Ist das der Fall, wird

der Benutzer automatisch auch dort angemeldet, ohne seine Zugangsdaten auch noch für die Anmeldung im SQL Server eingeben zu müssen.

Authentifizierung über ein SQL Server-Konto

Die zweite Alternative ist, bei der Installation des SQL Servers für die Authentifizierung den sogenannten gemischten Modus zu aktivieren. Dann ist nicht nur die Windows-Authentifizierung möglich, sondern auch die SQL Server-Authentifizierung. Das bedeutet, dass im SQL Server Konten für Benutzer und Benutzergruppen unabhängig von den Windows-Konten angelegt werden.

Der Hintergrund ist, dass es nicht immer eine Domäne gibt, unter der die Benutzer verwaltet werden und somit ein vereinfachter Zugriff auch auf die Daten im SQL Server ermöglicht werden kann. Vielleicht haben Sie nur einen einzelnen Rechner und wollen eine lokale SQL Server-Datenbank verwenden, haben ein Netzwerk ohne Domäne, in dem die Benutzer aber dennoch über das Netz auf die Daten von SQL Server-Datenbanken zugreifen sollen oder Sie verwenden den SQL Server für eine Webanwendung.

Die für die SQL Server-Nutzung angelegten Konten sind also unabhängig von Windows-Konten. Somit benötigen wir also auch ein komplett eigenständiges Paar aus Benutzername und Kennwort. Wenn der Benutzer sich nun am SQL Server anmelden möchte, muss er unter Umständen zwei Anmeldungen durchführen – erst am Windows-System und dann am SQL Server.

Übrigens: Das Benutzerkonto **sa**, das standardmäßig vorhanden ist, ist auch ein SQL Server-Konto. Unnötig zu erwähnen, dass dieses Konto immer mit einem Kennwort geschützt werden muss, da dieses universellen Zugriff auf den SQL Server bietet. Es gibt eine Bedingung, unter der sich ein Benutzer auch ohne Verwendung einer Domäne per Windows Authentifizierung am SQL Server anmelden kann – wenn er sich auf dem gleichen Rechner angemeldet hat, auf dem auch der SQL Server installiert ist.

Zweistufige Sicherheit

Genau so, wie der Benutzer sich erst am Windows-System und dann am SQL Server anmelden muss (wobei letzteres automatisch geschieht, wenn die Windows-Authentifizierung verwendet wird), gibt es auch innerhalb des SQL Servers zwei Stufen der Absicherung.

Wenn ein Benutzer ein Konto am SQL Server besitzt, heißt das nämlich noch lange nicht, dass er sich auch an einer Datenbank im SQL Server anmelden kann. Das Benutzerkonto muss auch für die Datenbank eingetragen sein und kann dort dann mit spezifischen Rechten versehen werden.

Im Prinzip erfolgt die Anmeldung an einer Datenbank im SQL Server also so, dass der SQL Server erst prüft, ob es überhaupt ein Benutzerkonto mit den angegebenen Daten im SQL Server gibt. Ist das nicht der Fall, scheitert die Anmeldung bereits in der ersten Stufe. Gibt es ein Benutzerkonto und ist das Kennwort korrekt beziehungsweise handelt es sich um eine Windows-Authentifizierung, prüft der SQL Server als Nächstes, ob das Benutzerkonto auch der jeweiligen Datenbank zugeordnet ist.

Benutzer und Benutzergruppen

Wie auch in einer Windows-Domäne gibt es auch beim SQL Server Benutzer und Benutzergruppen. Beiden können Sie Berechtigungen für den Zugriff auf die verschiedenen Elemente einer Datenbank zuweisen.

Der Vorteil der Konten von Benutzergruppen gegenüber Benutzern treten dabei zutage, wenn mehrere Benutzer mit gleichen Berechtigungen für den Zugriff auf die Datenbank ausgestattet werden sollen. Dann brauchen Sie die Berechtigungen nur für die Benutzergruppe zu definieren und brauchen die Benutzer, die diese Berechtigungen erhalten sollen, einfach nur der Benutzergruppe zuzuweisen.

Das hat nebenher den Vorteil, dass Sie, wenn ein Benutzer ausscheidet und/oder ein neuer Benutzer die gleichen Berechtigungen erhalten soll, Sie den Benutzer nur von der

Gruppe entfernen oder hinzufügen müssen. Andersherum brauchen Sie, wenn sich einmal die Berechtigungen für die Benutzer dieser Gruppe ändern sollen, nur die Berechtigungen für die Gruppe zu ändern und nicht für jeden einzelnen Benutzer. Sie sehen: Die Vergabe von Berechtigungen über Benutzergruppen ist essenziell, wenn Sie es mit mehr als einem Benutzer zu tun haben.

Bei der Verwendung der Windows-Authentifizierung gibt es gegebenenfalls bereits Benutzergruppen unter Windows, denen die verschiedenen Benutzer zugeordnet sind. In diesem Fall vergeben Sie die Berechtigungen im SQL Server für die Windows-Benutzergruppen und nicht für einzelne Benutzer. In Unternehmen sollten dies der Fall sein – dort gibt es verschiedenen Benutzergruppen mit individuellen Berechtigungen, denen die Mitarbeiter zugeordnet werden. Dadurch erben die Mitarbeiter die Berechtigungen dieser Gruppe. Scheidet ein Mitarbeiter aus dem Unternehmen aus, wird sein Benutzerkonto aus der Benutzergruppe entfernt – schon sind auch seine Berechtigungen am SQL Server erloschen. Auf diese Weise muss sich der Administrator des SQL Servers oder der Datenbank gar nicht um die Vergabe oder den Entzug von Berechtigungen beschäftigen, denn dies erledigt der Systemadministrator, indem er die Windows-Benutzer aus Gruppen entfernt oder sie diesen hinzufügt.

Wenn Sie Berechtigungen für Benutzergruppen mit Benutzern in einer Umgebung mit SQL Server-Authentifizierung vergeben wollen, müssen Sie die Benutzergruppen erst noch im SQL Server anlegen und diesen die einzelnen Benutzer zuweisen. Das ist, wie oben erwähnt, die einzig mögliche Variante, wenn die Benutzer und Benutzergruppen nicht in einer Domäne verwaltet werden.

Sicherheit mit Windows-Authentifizierung

Wir schauen uns als Erstes an, wie die Windows-Authentifizierung funktioniert. Dazu legen wir im Folgenden auf dem lokalen Rechner zwei Windows-Benutzerkonten an sowie zwei Benutzergruppen an. Jedes Benutzerkonto weisen wir dabei einer Benutzergruppe zu. Dann fügen wir die Benutzergruppen als Anmeldungen zum SQL Server hinzu und ordnen diese unserer Beispieldatenbank **Suedsturm_SQL** zu.

Benutzer hinzufügen

Unter Windows 10 Pro können Sie Benutzer über die Benutzeroberfläche hinzufügen. Da es unter den Lesern gegebenenfalls auch Benutzer der Home-Version von Windows 2010 gibt, wollen wir einen alternativen Weg zum Verwalten von Benutzern und Benutzergruppen aufzeigen. Dieser verwendet das Kommandozeilentool **net.exe**. Um dieses zu nutzen, geben Sie **cmd** in das Suchfeld von Windows ein und betätigen die Eingabetaste. In der nun erscheinenden Eingabeaufforderung können Sie mit wenigen Befehlen die Benutzer und Benutzergruppen verwalten. Die Befehle von **net.exe** finden Sie, indem Sie diesen einfach ohne weitere Optionen eingeben (siehe Bild 1).

Um sich einen Überblick über die aktuell vorhandenen Benutzer zu schaffen, verwenden Sie etwa den folgenden Befehl:

```
net.exe USER
```

Dies liefert die Liste aus Bild 2. Wenn Sie alle Benutzergruppen ausgeben möchten, ist die folgende Option die Richtige:

```
net.exe GROUP
```

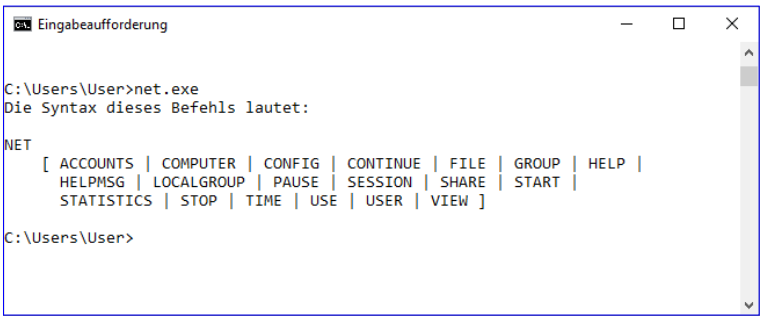


Bild 1: Optionen des Befehls **net.exe** zur Benutzerverwaltung

Dieser Befehl liefert, wenn keine Domäne vorhanden ist, das folgende Resultat:

```
Dieser Befehl kann nur auf Windows-Domänencont-
rollern ausgeführt werden.
```

Wenn wir allerdings den Befehl **net.exe LOCALGROUP** ausführen, liefert dies das Ergebnis aus Bild 3.

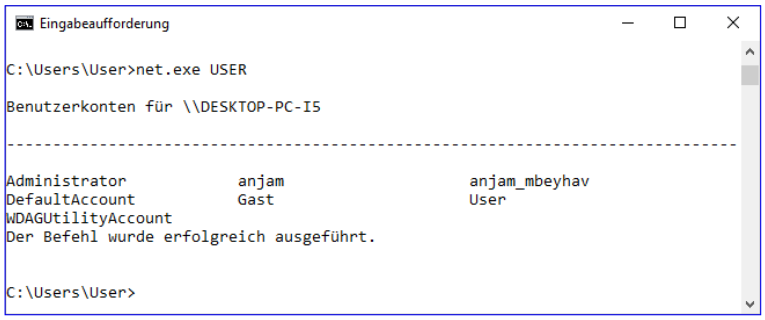


Bild 2: Ausgabe der aktuellen Benutzer des Systems

Wir wollen nun nicht lange experimentieren, sondern direkt einen neuen Benutzer anlegen. Um einen neuen Benutzer namens **Benutzer1** mit dem Kennwort **password** anzulegen, verwenden wir den folgenden Befehl:

```
net.exe USER /ADD Benutzer1 password
```

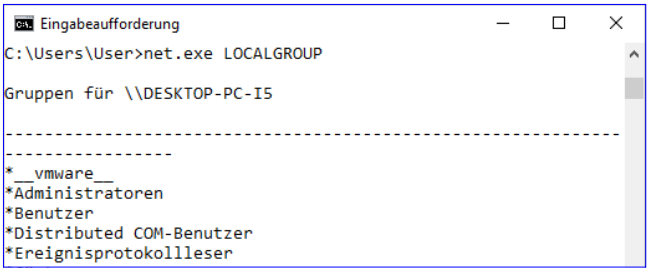


Bild 3: Ausgabe der Benutzergruppen

Dabei erhalten wir allerdings den folgenden Fehler:

```
C:\>net.exe USER /ADD Benutzer1
password
Systemfehler 5 aufgetreten.

Zugriff verweigert
```

Aber keine Sorge: Wir haben lediglich versäumt, die Eingabeaufforderung mit Administrator-Rechten zu starten. Dazu geben Sie nochmals **cmd** im Suchfeld von Windows ein. Es erscheint nun der Eintrag Eingabeaufforderung in der Liste der Suchergebnisse.

Diesen klicken Sie mit der rechten Maustaste an und wählen aus dem Kontextmenü den Befehl **Als Administrator ausführen** aus (siehe Bild 4).

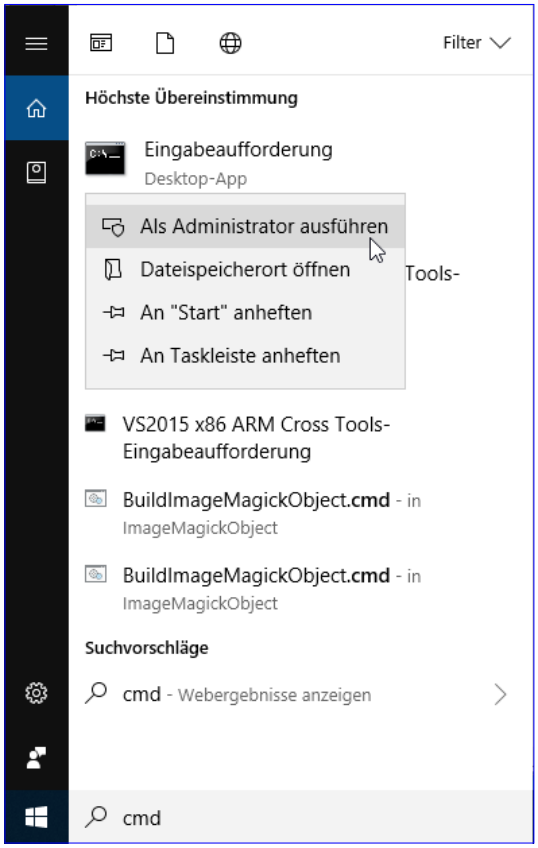


Bild 4: Eingabeaufforderung mit Administratorrechten starten

Danach gelingt dann auch der Aufruf des Befehls zum Anlegen eines neuen Benutzers:

```
C:\>net.exe USER /Add Benutzer1
password
Der Befehl wurde erfolgreich aus-
geführt.
```

Geben Sie danach erneut den Befehl **net.exe USER** ein, erscheint der neue Benutzer namens **Benutzer1** in der Auflistung. Auf die gleiche Weise legen wir nun den Benutzer mit dem Namen **Benutzer2** und dem Kennwort **password** an:

```
C:\>net.exe USER /Add Benutzer2
password
```

Damit haben wir die gewünschten beiden Benutzer zum System

hinzugefügt. Nun kommen wir zu den Benutzergruppen. Dazu finden wir zwei mögliche Optionen in der Befehlsliste: **GROUP** und **LOCALGROUP**. Der Befehl **net.exe GROUP** kann nur in Zusammenhang mit einer Domäne angewendet werden. Da wir diese nicht für jeden Leser voraussetzen können, wollen wir die Variante **LOCALGROUP** nutzen, um die neuen Gruppen anzulegen. Mit der Option **/ADD** und dem Namen der hinzuzufügenden Gruppe, hier **Gruppe1**, fügen wir nun eine erste Gruppe hinzu:

```
C:\>net.exe LOCALGROUP /ADD Gruppe1
Der Befehl wurde erfolgreich ausgeführt.
```

Auf diese Weise legen wir auch noch eine zweite Gruppe an:

```
C:\>net.exe LOCALGROUP /ADD Gruppe2
Der Befehl wurde erfolgreich ausgeführt.
```

Der Befehl **net.exe LOCALGROUP** gibt nun die bereits vorhandenen und die beiden neuen Gruppen aus:

```
C:\>net.exe LOCALGROUP
Gruppen für \\DESKTOP-PC-I5
-------------------------------------------------------
...
*Gruppe1
*Gruppe2
...
Der Befehl wurde erfolgreich ausgeführt.
```

Benutzer einer Gruppe anzeigen

Bevor wir unsere beiden Benutzer den neuen Gruppen zuweisen, schauen wir uns an, wie wir ermitteln können, welche Benutzer einer Gruppe bereits zugeordnet sind. Dazu verwenden wir den Befehl **LOCALGROUP** und geben einfach den Namen der Gruppe an, deren Mitglieder wir ermitteln wollen – hier für die Gruppe **Administratoren**:

```
C:\WINDOWS\system32>net.exe LOCALGROUP Administratoren
Aliasname        Administratoren
Beschreibung     Administratoren haben uneingeschränkten
Vollzugriff auf den Computer bzw. die Domäne.
Mitglieder
-------------------------------------------------------
Administrator
User
Der Befehl wurde erfolgreich ausgeführt.
```

Benutzer einer Gruppe hinzufügen

Nun fügen wir den Benutzer mit den Namen **Benutzer1** der Gruppe namens **Gruppe1** hinzu. Dazu nutzen wir wieder den Befehl **LOCALGROUP** mit den Namen der Gruppe, fügen aber noch die Option **/ADD** mit den Namen des hinzuzufügenden Benutzers hinzu:

```
C:\>net.exe LOCALGROUP Gruppe1 /ADD Benutzer1
Der Befehl wurde erfolgreich ausgeführt.
```

Ein Aufruf des Befehls **LOCALGROUP Gruppe1** liefert dann den Benutzer **Benutzer1** als einziges Mitglied dieser Gruppe.

Den Benutzer **Benutzer2** fügen wir anschließend auf die gleiche Weise der Gruppe **Gruppe2** hinzu.

Benutzer aus einer Gruppe entfernen

Wenn Sie den Benutzer anschließend wieder aus der Gruppe entfernen wollen, erledigen Sie das mit der folgenden Anweisung:

```
c:/>LOCALGROUP Gruppe1 /DELETE Benutzer1
```

Um im Anschluss an eventuelle Experimente im Rahmen der Lektüre dieses Artikels das System in den ursprünglichen Zustand zu versetzen, können Sie die Benutzer wie folgt löschen:

```
C:\>net.exe LOCALGROUP /DELETE Gruppe1
C:\>net.exe LOCALGROUP /DELETE Gruppe2
```

Die beiden Benutzer löschen Sie so:

```
C:\>net.exe USER /DELETE Benutzer1
C:\>net.exe USER /DELETE Benutzer2
```

Für die folgenden Experimente im SQL Server sollten Sie die beiden Gruppen und die beiden Benutzer jedoch noch behalten beziehungsweise erneut anlegen.

Benutzergruppe im SQL Server nutzen

Nun wollen wir die beiden zuvor unter Windows angelegten Benutzergruppen im SQL Server nutzen. Dazu öffnen Sie das SQL Server Management Studio, in unserem Fall in der Version 2016. Hier arbeiten wir mit einer Datenbank namens **Suedsturm_SQL**, die Sie gegebenenfalls noch anlegen müssen. Das benötigte SQL-Skript finden Sie im Download zu diesem Beitrag, die Anleitung dazu liefert der Beitrag **SQL Server-Datenbank kopieren** (**www.access-im-unternehmen.de/1153**).

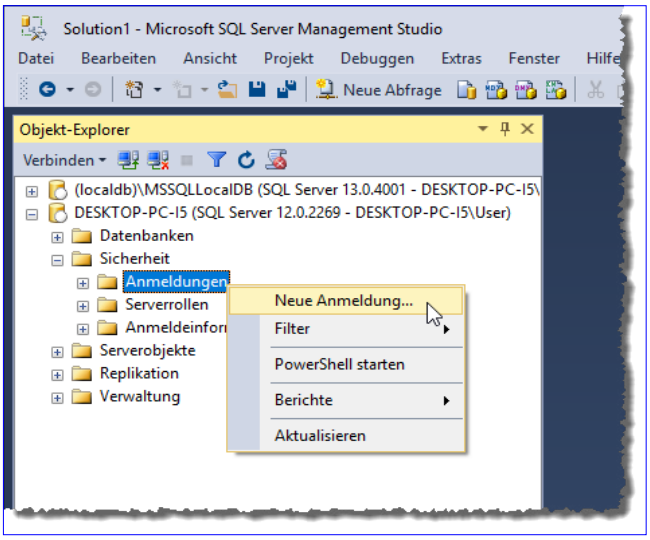


Bild 5: Aufruf des Dialogs zum Anlegen einer neuen Anmeldung

Zugriff einrichten in zwei Stufen

Wie wir weiter oben angeführt haben, hat das Sicherheitssystem des SQL Servers zwei Stufen – die erste erlaubt den Zugriff auf den SQL Server, die zweite dann den Zugriff auf die jeweilige Datenbank. Dementsprechend müssen wir auch zuerst entsprechende Benutzergruppen und Benutzer für den SQL Server einrichten und erst dann für die jeweilige Datenbank.

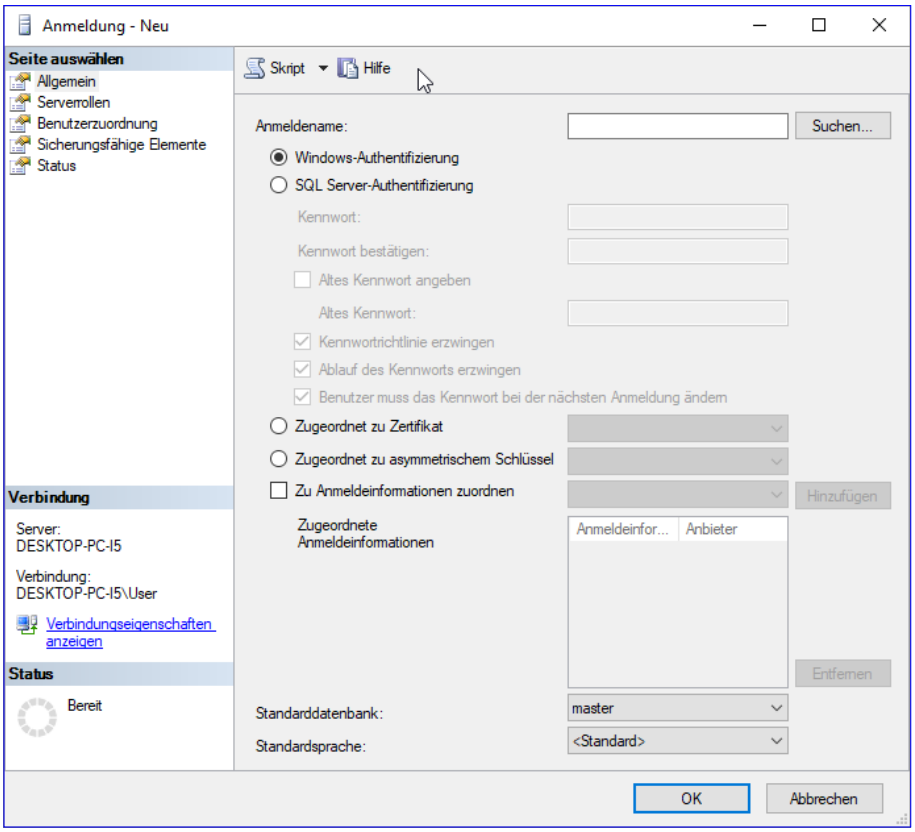


Bild 6: Dialog zum Anlegen einer neuen Anmeldung

Um zunächst eine neue Gruppe für die Instanz des SQL Servers anzulegen, klicken Sie mit der rechten Maus auf den Eintrag **Sicherheit** des SQL Servers im SQL Server Management Studio und wählen den Eintrag **Neue Anmeldung...** aus (siehe Bild 5).

Es erscheint der Dialog aus Bild 6. Hier können Sie den Namen der anzulegenden Gruppe direkt in das Feld **Anmeldename** eingeben. Allerdings wollen wir es uns

etwas gemütlicher machen und klicken zunächst auf die Schaltfläche **Suchen**.

Dies öffnet wiederum den Dialog aus Bild 7. Auch hier wollen wir noch nicht nach der anzulegenden Gruppe suchen, die wir ja von den Windows-Anmeldungen übernehmen wollen. Wir klicken auf die Schaltfläche **Erweitert**.

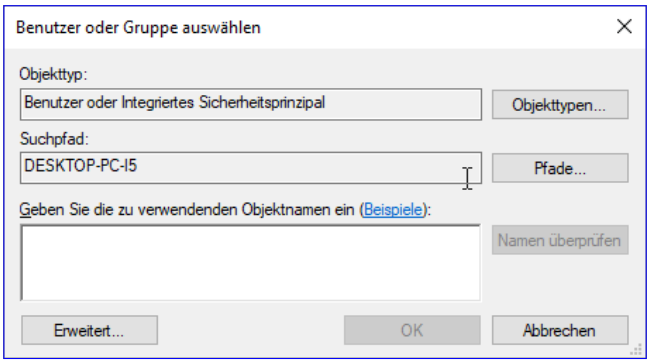


Bild 7: Dialog zum Auswählen von Benutzern oder Gruppen

Im nun erscheinenden Dialog **Benutzer oder Gruppe auswählen** klicken wir ohne Angabe von Suchkritereien einfach auf die Schaltfläche **Jetzt suchen**. Es erscheinen einige Einträge in der Liste der Suchergebnisse, darunter auch unsere zuvor angelegten Benutzer namens **Benutzer1** und **Benutzer2** (siehe Bild 8). Allerdings finden wir hier leider nicht die beiden Gruppen **Gruppe1** und **Gruppe2**.

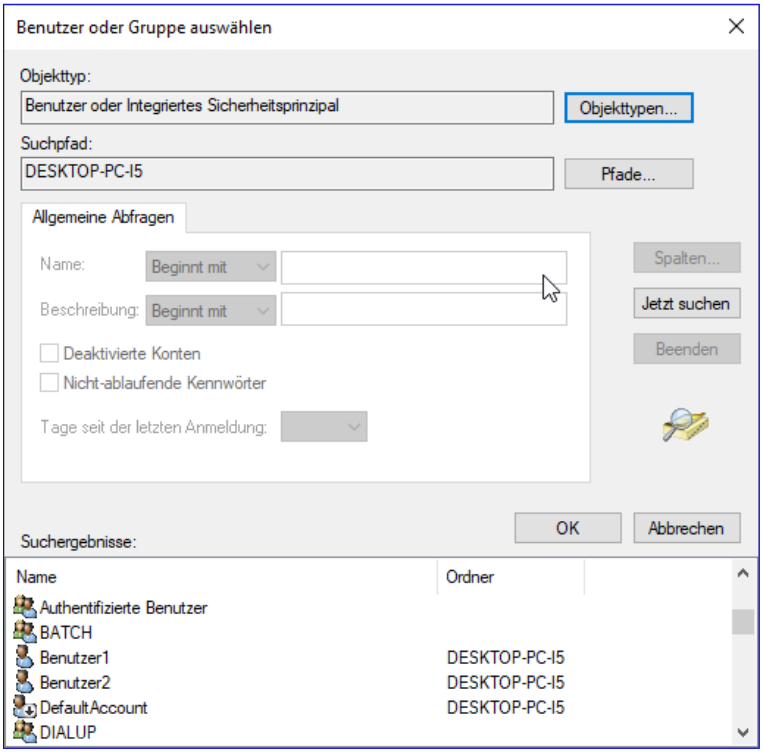


Bild 8: Erweiterter Dialog zum Auswählen von Benutzern oder Gruppen

Die Ursache ist schnell gefunden. Wenn Sie auf die Schaltfläche **Objekttypen** rechts neben dem Textfeld Objekttyp klicken, erscheint der Dialog **Objekttypen**. Hier ist die Option **Gruppen** standardmäßig deaktiviert. Also setzen wir den fehlenden Haken und schließen den Dialog wieder (siehe Bild 9).

Zurück im Dialog **Benutzer oder Gruppe auswählen** klicken Sie nun erneut auf die Schaltfläche **Jetzt suchen**. Nun erscheinen in der Liste der Suchergebnisse auch die beiden Einträge **Gruppe1** und **Gruppe2** (siehe Bild 10). Hier markieren wir nun zunächst den Eintrag **Gruppe1** und klicken auf **OK**.

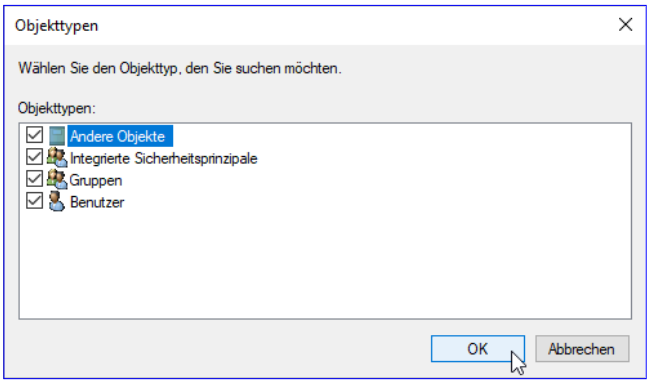


Bild 9: Dialog zur Auswahl der anzuzeigenden Objekttypen

Im Dialog **Benutzer oder Gruppe auswählen** finden Sie nun den Eintrag mit dem Rechnernamen und dem Namen der Gruppe (siehe Bild 11).

Dieser wird nach dem Schließen des Dialogs in das Feld Anmeldename des Dialogs **Anmeldename - Neu** übernommen. Wenn Sie den Rechnernamen und den Namen

der Gruppe kennen, können Sie diesen natürlich auch hier eintippen – anderenfalls gehen Sie den zuvor beschriebenen Weg über die Suchdialoge. In diesem Dialog stellen Sie unten für die Eigenschaft **Standarddatenbank** noch den Namen der Datenbank ein, auf die die Benutzergruppe zugreifen soll – in unserem Beispiel **Suedsturm_SQL** (siehe Bild 12).

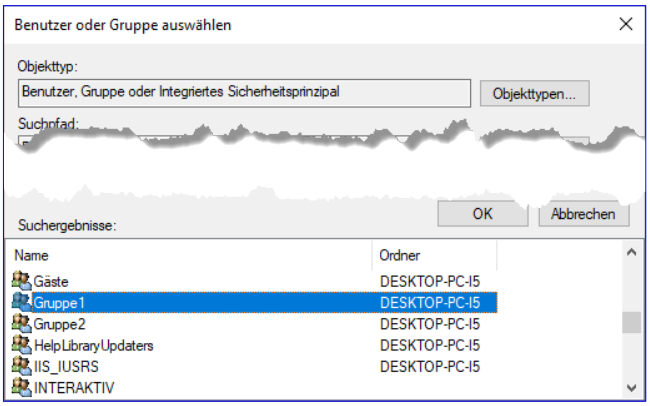


Bild 10: Dialog mit den gesuchten Gruppen

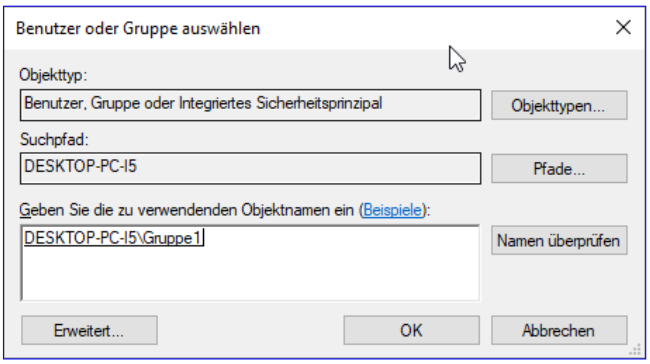


Bild 11: Die gesuchte Gruppe im Dialog **Benutzer oder Gruppe auswählen**

Serverrolle einstellen

Danach wechseln wir zur zweiten Seite des Dialogs namens **Serverrollen**. Hier stellen wir ein, welche Berechtigungen die Benutzergruppe erhalten soll. Die Serverrolle **public** ist bereits voreingestellt (siehe Bild 13). Die übrigen Serverrollen haben folgende Bedeutung:

- **bulkadmin**: Ausführen von Massenimporten
- **dbcreator**: Anlegen, ändern, löschen und wiederherstellen von Datenbanken
- **diskadmin**: Verwalten von Datensicherungsmedien
- **processadmin**: Beenden von Prozessen
- **public**: Standardserverrolle
- **securityadmin**: Verwalten von Anmeldungen
- **serveradmin**: Ändern der SQL Server-Konfiguration, sowie Beenden vom SQL Server-Dienst
- **setupadmin**: Hinzufügen und entfernen von Verbindungsservern
- **sysadmin**: Ausführen aller Aktivitäten im SQL Server

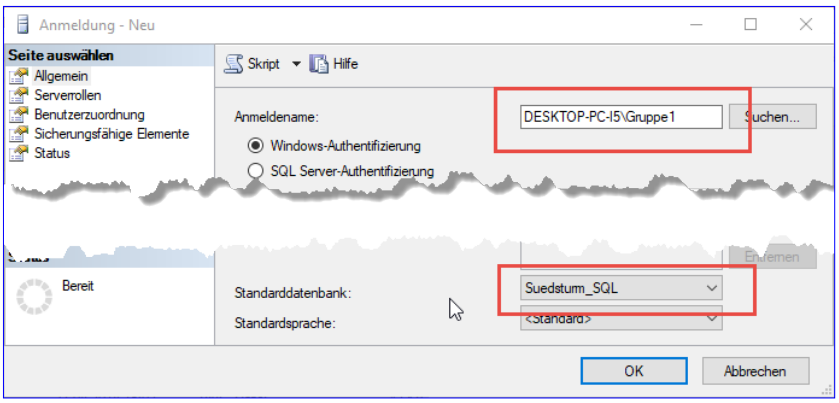


Bild 12: Die gesuchte Gruppe im Dialog **Benutzer oder Gruppe auswählen**

Wenn Sie genauer wissen wollen, welche Berechtigungen sich hinter den einzelnen Serverrollen befinden, können Sie diese mit einer SQL-Abfrage ermitteln. Dazu öffnen Sie eine neue Abfrage über den Kontextmenü-Eintrag **Neue Abfrage...** des Server-Eintrags im Objekt-Explorer und geben den folgenden Befehl ein:

```
exec sp_srvrolepermission <Serverrolle>
```

SQL Server-Authentifizierung

Im Beitrag »SQL Server: Sicherheit und Benutzerverwaltung« haben wir gezeigt, wie Sie grundsätzlich mit Benutzern, Benutzergruppen und den Berechtigungen an den einzelnen Elementen einer SQL Server-Datenbank und dem SQL Server selbst umgehen. Dort sind wir allerdings nur auf den Zugriff per Windows-Authentifizierung eingegangen. Viele Szenarios erfordern es allerdings, dass Sie die SQL Server-Authentifizierung verwenden, was bedeutet, dass Sie unter dem SQL Server eigene Konten für Benutzergruppen und Benutzer anlegen – unabhängig von den Windows-Benutzergruppen und -Benutzern. Dieser Beitrag liefert die Grundlagen rund um die Verwendung der SQL Server-Authentifizierung.

Voraussetzung

Für die Beispiele in diesem Artikel benötigen Sie die Datenbank **Suedsturm_SQL**, die Sie im Download zu diesem Artikel finden und wie im Beitrag **SQL Server-Datenbank kopieren** (**www.access-im-unternehmen.de/1153**) beschrieben installieren können. Dazu benötigen Sie außerdem eine SQL Server-Instanz auf Ihrem Rechner sowie das SQL Server Management Studio.

Warum SQL Server-Authentifizierung?

Die im Beitrag **SQL Server: Sicherheit und Benutzerverwaltung** (**www.access-im-unternehmen.de/1154**) beschriebene Windows-Authentifizierung setzt voraus, dass es für jedes im SQL Server eingerichtete Konto, sei es ein Benutzer- oder ein Benutzergruppenkonto, ein entsprechendes Konto unter Windows eingerichtet ist. Das ist in Umgebungen, wo der SQL Server auf der gleichen Maschine arbeitet, auf welcher der Benutzer angemeldet ist, ohne Problem möglich. Soll jedoch auch über das Netzwerk auf die Datenbank zugegriffen werden, gelingt dies unter der Windows-Authentifizierung nur, wenn der Zugriff im Kontext einer Windows-Domäne erfolgt. Dies ist jedoch nicht immer möglich. Keinesfalls ist dies der Fall, wenn der Zugriff etwa über das Internet erfolgt – dann und in anderen Fällen gibt es allerdings eine Alternative.

Dabei handelt es sich um die sogenannte SQL Server-Authentifizierung. Hier werden auf dem SQL Server von den Windows-Benutzern und -Benutzergruppen unabhängige Benutzer und Benutzergruppen angelegt (Sie kommen natürlich auch nur mit Benutzern aus). Im Unterschied zur Windows-Authentifizierung, wo es ausreicht, für die Mitglieder einer Windows-Gruppe eine entsprechende Anmeldung für die Gruppe im SQL Server einzurichten, müssen Sie bei der SQL Server-Authentifizierung für jeden Benutzer ein Konto anlegen. Sie können natürlich zusätzlich zum Benutzer auch Benutzergruppen anlegen und die Berechtigungen dann je nach Anforderung für den Benutzer oder auch für die Benutzergruppe eines Benutzers definieren.

Zweistufige Sicherheit

Wie auch bei der Windows-Authentifizierung bietet auch die SQL Server-Authentifizierung ein zweistufiges Sicherheitsverfahren an, bei dem der Benutzer zunächst über die Berechtigung für die Anmeldung am SQL Server und dann noch über die Berechtigungen für die gewünschte Datenbank verfügen muss.

Benutzergruppe im SQL Server anlegen

Wenn Sie eine neue Benutzergruppe im SQL Server anlegen möchten, öffnen Sie über den Kontextmenü-Eintrag **Sicherheit|Neu|Anmeldung...** den Dialog **Anmeldung – Neu** (siehe Bild 1). Hier aktivieren Sie zunächst die Option **SQL Server-Authentifizierung**. Daran, dass nun die Schaltflächen Suchen neben dem Feld **Anmel-**

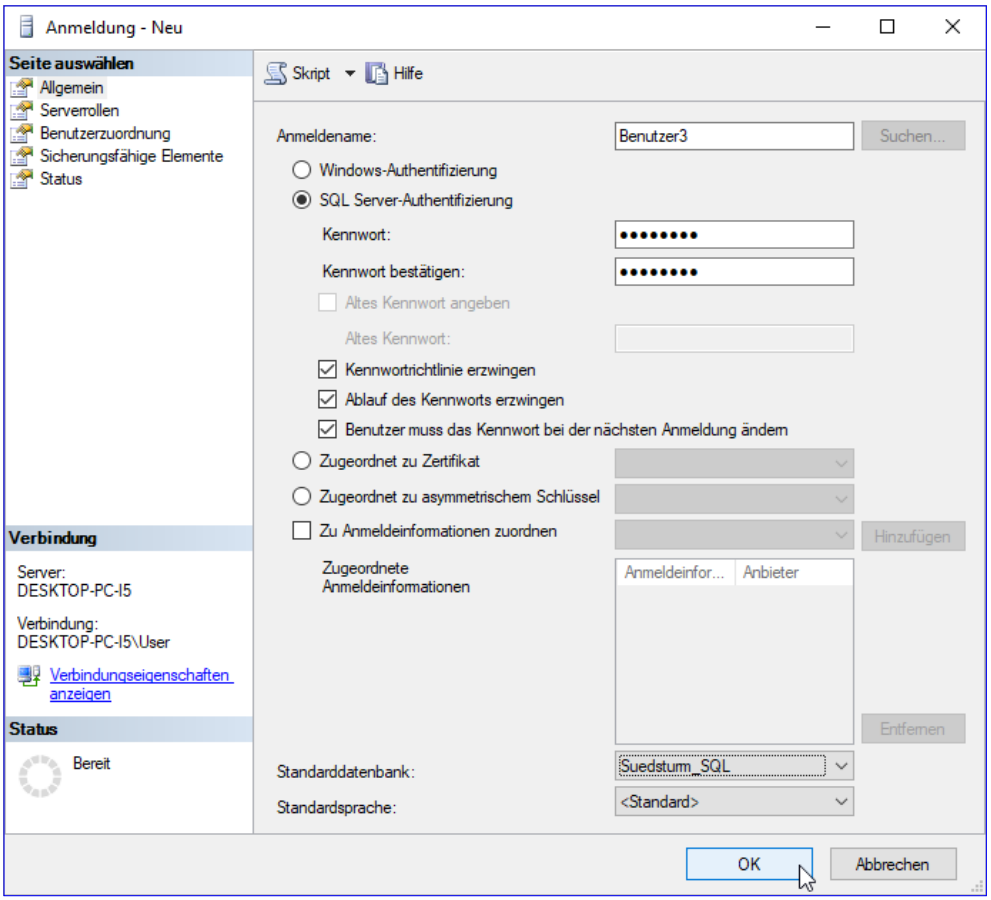


Bild 1: Anlegen eines neuen Benutzers unter der SQL Server-Authentifizierung

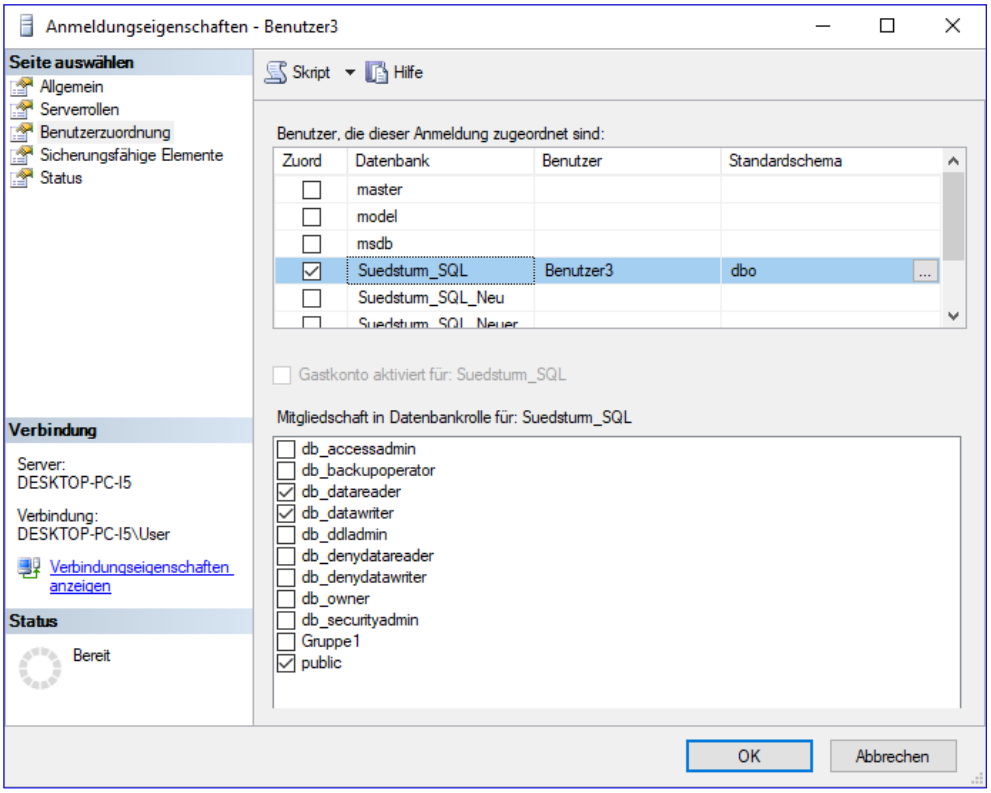


Bild 2: Zuweisen der Datenbank und der Berechtigungen für diese Datenbank

dename verschwindet, erkennen Sie schon, dass wir hier nicht mehr auf die vorhandenen Windows-Benutzerkonten zugreifen können. Also geben wir hier einen benutzerdefinierten Anmeldenamen ein. Da wir im Beitrag **SQL Server: Sicherheit und Benutzerverwaltung** bereits die Benutzer namens **Benutzer1** und **Benutzer2** verwendet haben, nutzen wir nun zunächst **Benutzer3**. Unter der Option **SQL Server-Authentifizierung** geben Sie außerdem das Kennwort ein – hier haben wir schlicht **password** verwendet. Unten wählen wir wieder die Zieldatenbank als Standarddatenbank aus, also **Suedsturm_SQL**. Auf der nächsten Seite unter Serverrollen behalten wir die Serverrolle **public** bei, da der Benutzer keine weiteren Rechte für die Administration des SQL Servers erhalten soll.

Auf der Seite **Benutzerzuordnung** wählen wir schließlich die Datenbank **Suedsturm_SQL** aus, auf die der Benutzer zugreifen können soll (siehe Bild 2). Daraufhin erscheinen im unteren Bereich die verschiedenen allgemeinen

Datenbankrollen für die Datenbank **Suedsturm_SQL**. Hier wählen wir zusätzlich zu **public** noch **db_datareader** und **db_datawriter** aus, damit der Benutzer in allen Tabellen lesen und schreiben darf.

Nur Benutzer, keine Gruppen

Hier wird auch gleich ein entscheidender Unterschied zwischen der Windows-Authentifizierung und der SQL Server-Authentifizierung deutlich: Bei der Windows-Authentifizierung können Sie sowohl Windows-Benutzer als auch Windows-Benutzergruppen mit dem Suchen-Dialog auffinden und als Anmeldungen anlegen.

Wenn Sie jedoch die Option **SQL Server-Authentifizierung** wählen, verschwindet mit dem **Suchen**-Button die Möglichkeit, Gruppen hinzuzufügen – und ein neuer Anmeldename ist bei der SQL Server-Authentifzierung immer ein neuer Benutzer, aber niemals eine Benutzergruppe.

Damit erhalten wir auch den Nachteil, dass wir bei SQL Server-Authentifizierung nicht den Vorzug genießen, Berechtigungen gleich für eine ganze Gruppe von Benutzern anlegen zu können.

Benutzerberechtigungen für eine Datenbank

Nachdem Sie die Anmeldung für den Benutzer für die SQL Server-Instanz erstellt haben, benötigen wir noch die zweite Stufe. Dazu wählen Sie aus dem Kontextmenü des Eintrags **Datenbanken\|Suedsturm_SQL\|Sicherheit** den Befehl **Neu\|Benutzer...** aus (siehe Bild 3). Alternativ können Sie auch den Kontextmenü-Befehl **Neuer Benutzer...** des Eintrags **...\|Sicherheit\|Benutzer** wählen.

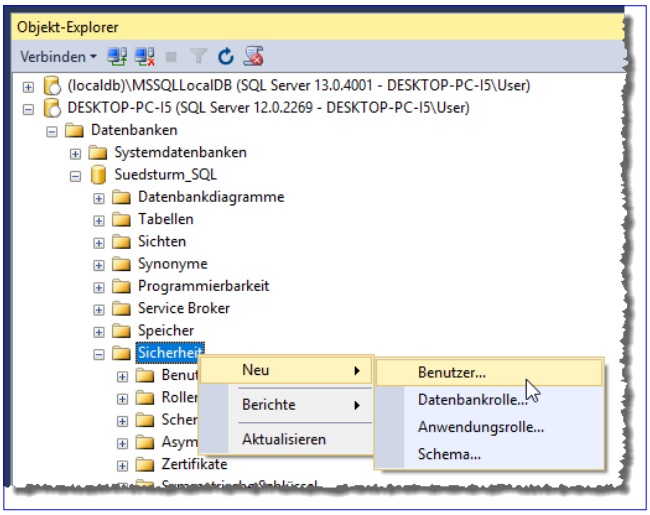


Bild 3: Hinzufügen des neuen Benutzers für die Datenbank

Aber Moment: Haben wir nicht weiter oben schon ... doch, wir haben: Auf der Seite **Benutzerzuordnung** des Dialogs **Anmeldung – Neu** haben wir bereits die Datenbank **Suedsturm_SQL** ausgewählt. Ein Blick auf die Liste unterhalb des Eintrags **Datenbanken\|Sicherheit\|Benutzer** liefert dann auch den Benutzer namens **Benutzer3** (siehe Bild 4).

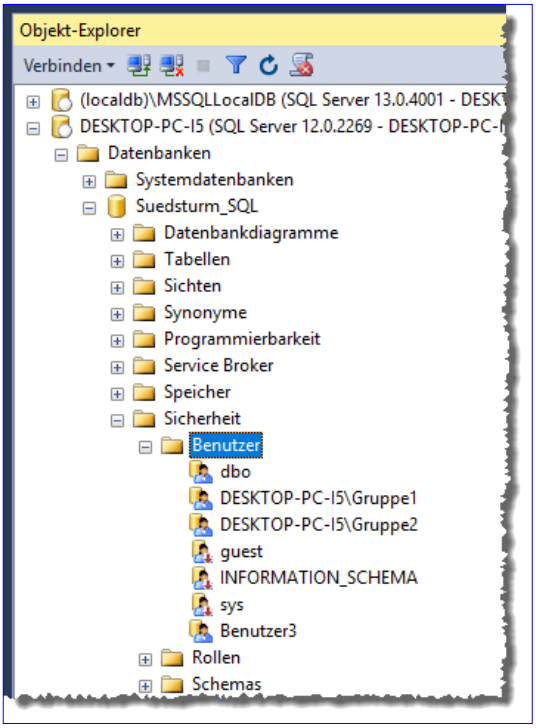


Bild 4: Der Benutzer namens **Benutzer3** ist schon in der Datenbank vorhanden.

Und weiter oben haben wir auch bereits die Berechtigungen für diesen Benutzer für die Datenbank **Suedsturm_SQL** eingerichtet.

Zugriff auf die Datenbank per SQL Server Management Studio

Nun wollen wir prüfen, wie wir unter einem anderen Benutzer auf die Datenbank zugreifen können, wenn wir dies über das SQL Server Management Studio erledigen wollen. Da wir aktuell als Administrator eingeloggt sind, legen wir eine neue Verbindung zur SQL Server-Instanz an. Dazu klicken Sie oben im Objekt-Explorer auf den Eintrag

SQL Server-Zugriff ohne gespeichertes Kennwort

Wenn Sie per ODBC auf die Tabellen etwa eines SQL Servers zugreifen, kennen Sie das Dilemma: Entweder Sie nutzen die Windows-Authentifizierung, was nicht überall möglich ist, aber ohne größere Probleme funktioniert. Oder Sie verwenden die SQL Server-Authentifizierung. Dann haben Sie zwei Möglichkeiten: Entweder Sie erlauben das Speichern des Kennworts in den Systemtabellen von Access. Oder der Benutzer muss nach dem eigentlichen Anmelden nochmal seine Benutzerdaten eingeben, wenn der erste Zugriff auf eine Tabelle erfolgt. In dieser Lösung schauen wir uns diese Varianten an – und eine bessere, bei der Sie einfach die Tabellen beim Start neu verknüpfen.

Wir gehen an dieser Stelle davon aus, dass Sie einen SQL Server auf dem Rechner installiert haben und dort eine Datenbank mit einer oder mehreren gewünschten Tabellen mit Zugriffsrechten für ein bestimmtes Konto eingerichtet haben. Wichtig ist letzteres – sonst können Sie ja nicht per SQL Server-Authentifizierung auf die Tabelle zugreifen.

Verknüpfung per Assistent

Der gängige Weg, eine Verknüpfung zu einer SQL Server-Tabelle herzustellen, ist der über den Assistenten. Dabei rufen Sie diesen über den Ribbon-Eintrag **Externe Daten|Importieren und Verknüpfen|Neue Datenquelle|Aus Datenbank|Aus SQL Server** auf (siehe Bild 1).

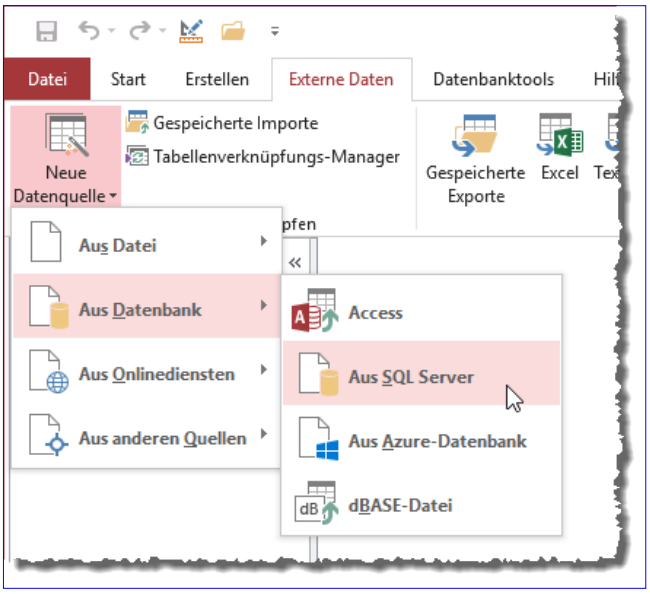


Bild 1: Manuelles Verknüpfen mit einer SQL Server-Tabelle

Dies öffnet den Dialog aus Bild 2, wo Sie die zweite Option zum Erstellen einer Verknüpfung auswählen und dann auf **OK** klicken.

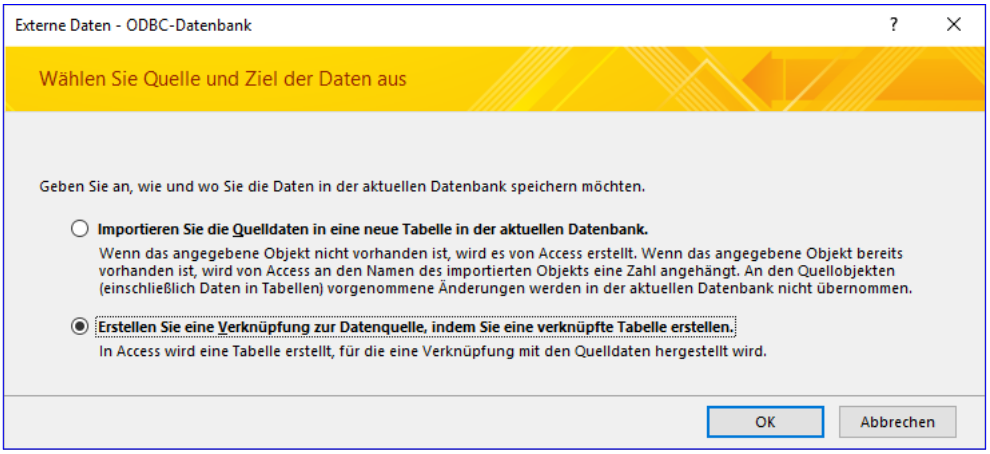


Bild 2: Start der Erstellung einer Verknüpfung

Der nächste Dialog heißt Datenquelle auswählen und hat zwei Registerreiter.

Wir wählen den zweiten namens **Computerdatenquelle** an und klicken dort auf die Schaltfläche **Neu...** (siehe Bild 3).

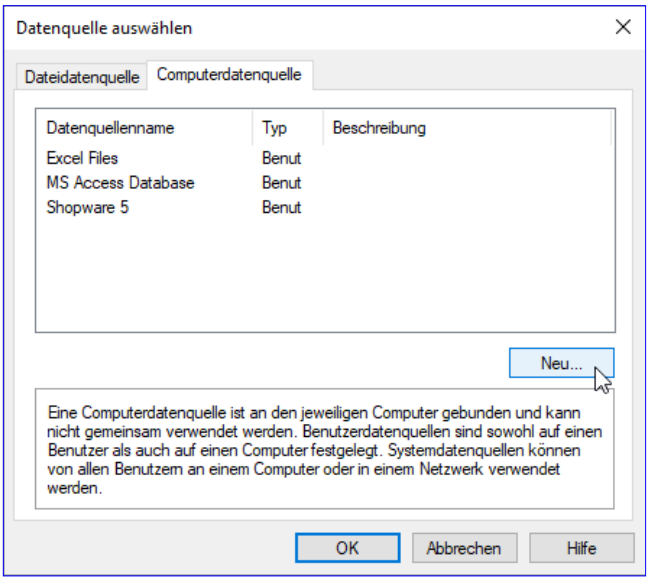


Bild 3: Anlegen einer neuen Computerdatenquelle

Einen Schritt weiter legen Sie noch genau fest, ob Sie eine Benutzerdatenquelle oder eine Systemdatenquelle anlegen wollen – wir wählen die Benutzerdatenquelle (siehe Bild 4).

Der folgende Schritt fragt ab, welchen Datenbanktreiber wir verwenden wollen. Mit dem Treiber **SQL Server** macht man nichts verkehrt (siehe Bild 5). **Die SQL Server Native Client x.0**-Treiber können Sie verwenden, wenn diese spezielle Features für den Zugriff auf die jeweilige Version des SQL Servers bereithalten.

Der nächste Schritt ermittelt einen Namen für die Datenquelle. Diese nennen wir einfach **Datenquelle_Suedsturm_SQL**. Außerdem möchte der Assistent hier wissen, auf welchen SQL Server wir zugreifen wollen, um die Verknüpfung herzustellen. Diese heißt in unserem Fall wie der Rechner, auf dem der SQL Server läuft. Wenn Sie die LocalDb-Version nutzen, würden Sie hier beispielsweise **(localdb)\SQLServerLocalDB** angeben (siehe Bild 6).

Ist dieser Schritt erledigt, fragt der Assistent im nächsten Schritt ab, ob wir Windows-Authentifizierung oder SQL Server-Authentifizierung verwenden wollen. Wir entscheiden uns für letzteres und tragen unsere Benutzerdaten

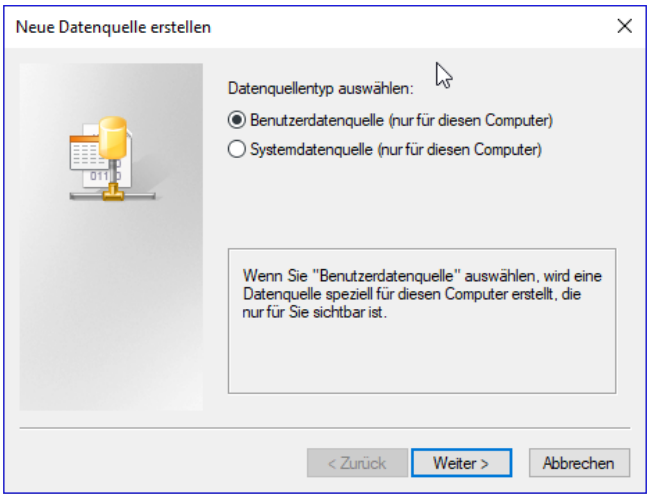


Bild 4: Auswahl des Datenquellentyps

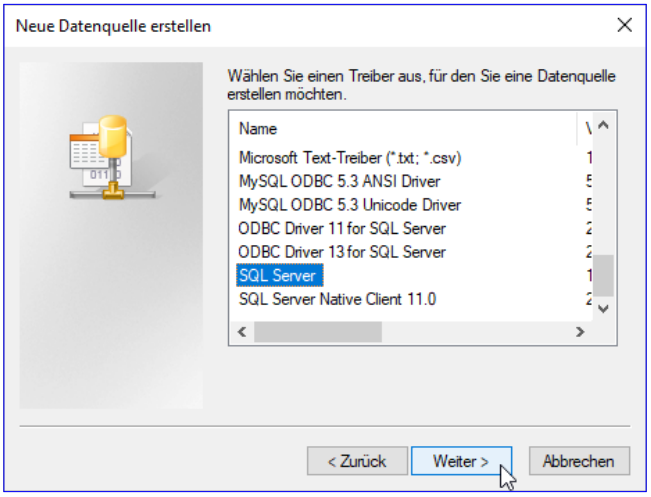


Bild 5: Auswahl des Datenbanktreibers

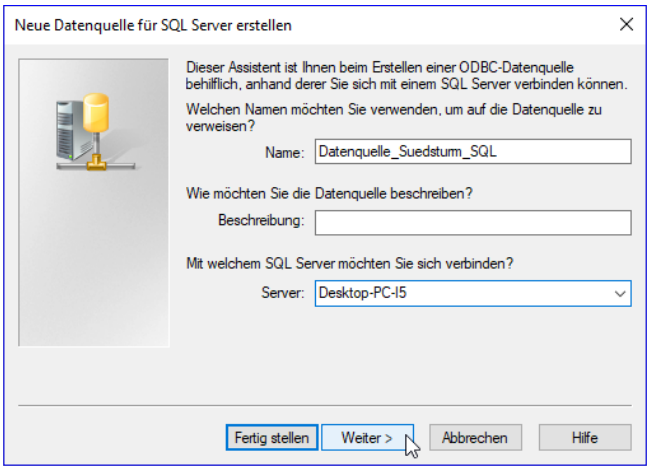


Bild 6: Angeben eines Namens für die Datenquelle und des Servers

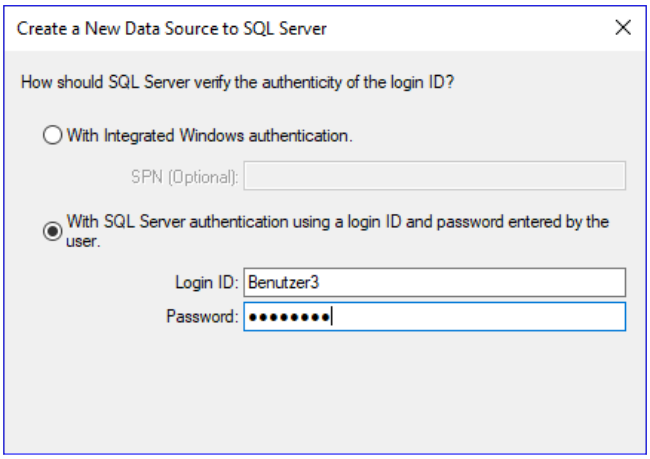


Bild 7: Einstellen der SQL Server-Authentifizierung und Angabe der Zugangsdaten

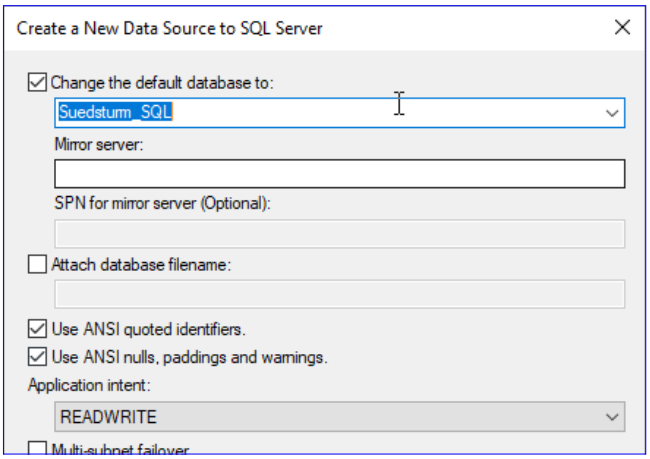


Bild 8: Einstellen der Standarddatenbank

in die beiden nun freigeschalteten Textfelder **Login ID** und **Password** ein (siehe Bild 7). Der Benutzer heißt hier **Benutzer3**, das Kennwort **password**.

Wie Sie einen Benutzer für die SQL Server-Authentifizierung für eine Datenbank anlegen, hier **Benutzer3**, erfahren Sie im Beitrag **SQL Server-Authentifizierung** (**www.access-im-unternehmen.de/1155**).

Im folgenden Schritt können Sie noch die Standarddatenbank auf **Suedsturm_SQL** einstellen (siehe Bild 8), die Einstellungen im darauffolgenden Dialog behalten wir bei.

Es folgt ein Abschlussdialog, der die Einstellungen nochmal zusammenfasst und eine Schaltfläche anbietet, mit der Sie die Datenquelle testen können (**Test Data Source...**). Diese klicken Sie an und sollten dann das Ergebnis aus Bild 9 erhalten.

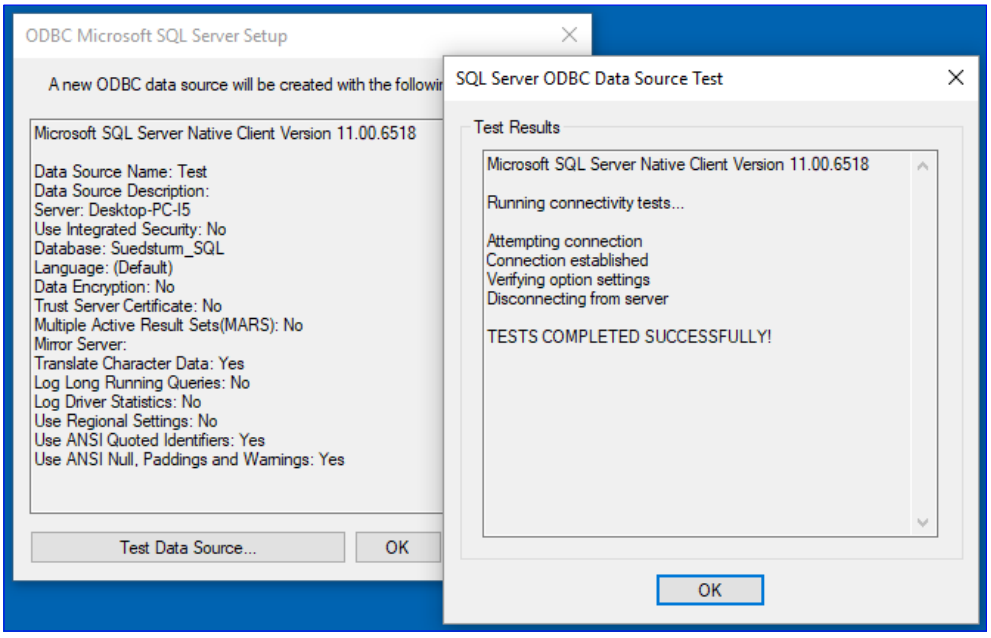


Bild 9: Testen der Datenbankverbindung

Damit ist die Computerdatenquelle angelegt und wir finden diese im Dialog **Datenquelle auswählen** vor. Die neue Datenquelle sollte nun ausgewählt sein und wir schließen den Dialog mit der **OK**-Schaltfläche.

Nun erscheint ein Dialog namens **SQL Server Login**. Hier ist der Benutzername bereits voreingestellt und wir müssen noch das Kennwort eingeben (siehe Bild 10). Mit einem Klick auf die Schaltfläche **Optionen >>** können Sie noch

Authentifizierung im SQL Server testen

Wenn Sie sich in die Programmierung des Sicherheitssystems des SQL Servers einarbeiten, wollen Sie die Berechtigungen der verschiedenen Windows-Benutzer, Windows-Gruppen oder SQL Server-Benutzer ausprobieren. Bei SQL Server-Benutzern, die sich über die SQL Server-Authentifizierung anmelden, ist das einfach. Bei Windows-Benutzern und -Benutzergruppen denkt der eine oder andere sicher schon darüber nach, dass er dann immer den Windows-Benutzer abmelden und sich unter dem Namen des zu prüfenden Benutzers einloggen muss. Das ist aber nicht der Fall – es gibt einfachere Methoden sowohl für den Zugriff von Access als auch direkt über das SQL Server Management Studio.

Testen mit SQL Server-Authentifizierung

Wir unterscheiden in diesem Beitrag wieder zwischen der SQL Server-Authentifizierung und der Windows-Authentifizierung. Bei der SQL Server-Authentifizierung ist es einfach. Wenn Sie einen Zugriff von Access aus ausführen wollen, um beispielsweise eine ODBC-Verknüpfung zu einer der SQL Server-Tabellen herzustellen, brauchen Sie ja nichts weiter zu tun als einfach die beim Herstellen der Verknüpfung verwendeten Zugangsdaten in der Verbindungszeichenfolge auf den jeweiligen Benutzer anzupassen.

In der Beispieldatenbank aus dem Beitrag **SQL Server-Zugriff ohne gespeichertes Kennwort** (**www.access-**

```
Private Sub cmdGo_Click()
    Dim strServer As String
    Dim strDatenbank As String
    Dim strVerbindungszeichenfolge As String
    DoCmd.OpenForm "frmLogin", windowMode:=acDialog
    If IstFormularGeoeffnet("frmLogin") Then
        strBenutzername = Nz(Forms!frmLogin!txtBenutzername, "")
        strKennwort = Nz(Forms!frmLogin!txtKennwort, "")
        DoCmd.Close acForm, "frmLogin"
        strServer = DLookup("Wert", "tblOptionen", "Bezeichnung = 'Server'")
        strDatenbank = DLookup("Wert", "tblOptionen", "Bezeichnung = 'Datenbank'")
        strVerbindungszeichenfolge = "ODBC;DRIVER={SQL Server};SERVER=" & strServer & ";DATABASE=" & strDatenbank _
            & ";UID=" & strBenutzername & ";PWD=" & strKennwort & ";OPTION=3;LOG_QUERY=1;"
        If VerbindungHerstellen(strVerbindungszeichenfolge) = True Then
            TabellenVerknuepfen strVerbindungszeichenfolge
            DoCmd.Close acForm, Me.Name
        Else
            'Aktionen beim Scheitern der Verknüpfung
        End If
    End If
End Sub
```

Listing 1: Prozedur zum Verknüpfen einer Tabelle

im-unternehmen.de/1156) verwenden wir beispielsweise die Prozedur aus Listing 1, um die im Formular frmLogin abgefragten Werte für Benutzername und Kennwort einzugeben. Hier können wir also beliebig im Kontext verschiedener Benutzer auf die Zieldatenbank zugreifen.

Ähnlich sieht es aus, wenn Sie die SQL Server-Authentifizierung im SQL Server Management Studio testen wollen. Sie können dann einfach jeweils eine neue Verbindung erstellen und sich als der entsprechende Benutzer einloggen. Dazu nutzen Sie den Befehl **VerbindenIDatenbankmodul** im Objekt-Explorer (siehe Bild 1).

Wenn Sie sich auf diese Weise angemeldet haben, arbeiten Sie im Kontext dieses Benutzers. Sie werden es dann schnell merken, wenn Sie versuchen, auf Elemente zuzugreifen, für die dieser Benutzer keine Berechtigungen hat.

Testen mit Windows-Authentifizierung im SQL Server Management Studio

Unter der Windows-Authentifizierung ist es ein wenig aufwendiger, den Zugriff unter einem anderen Benutzerkonto beziehungsweise unter einer anderen Benutzergruppe zu testen. Sie können ja, wie im Beitrag **SQL Server: Sicherheit und Benutzerverwaltung(www.access-im-unternehmen.de/1154**) beschrieben, auch eine Benutzergruppe statt eines Benutzers für die Anmeldung am SQL Server verwenden.

Wenn Sie nun mit Ihrem Standardbenutzerkonto am Windows-System angemeldet sind, können Sie beim Öffnen des SQL Server Management Studios und der Auswahl des Authentifizierungstyps Windows-Authentifizierung nicht den Windows-Benutzer auswählen, sondern können sich nur im Kontext des aktuell an Windows angemeldeten Benutzers anmelden. Deshalb sind die beiden Felder für Benutzername und Kennwort auch deaktiviert (siehe Bild 2).

Wie aber können wir nun eine Verbindung zum SQL Server als ein anderer Benutzer realisieren? Dazu klicken Sie mit der rechten Maustaste etwa auf den Eintrag für das SQL Server Management Studio. Hier erscheint ein weiterer Eintrag mit diesem Namen. Wenn Sie nun mit der rechten Maustaste auf diesen neuen Eintrag klicken, diesmal allerdings bei gedrückter Umschalt-Taste, erscheint ein

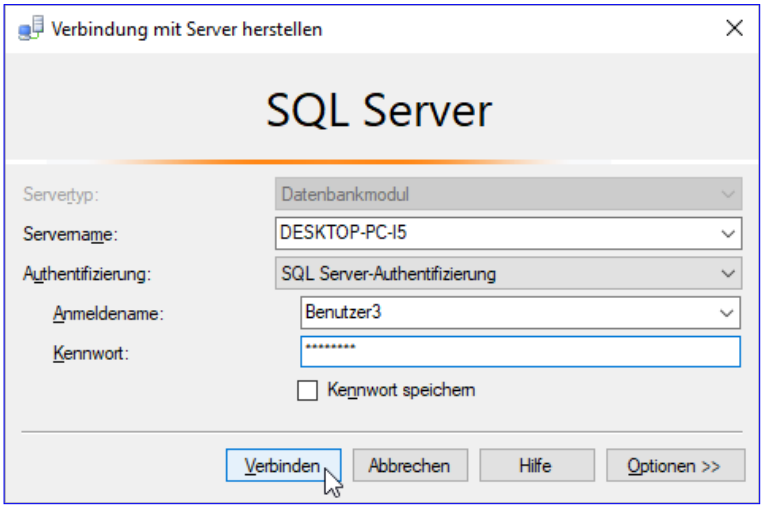


Bild 1: Neue Verbindung mit einem alternativen SQL Server-Benutzer

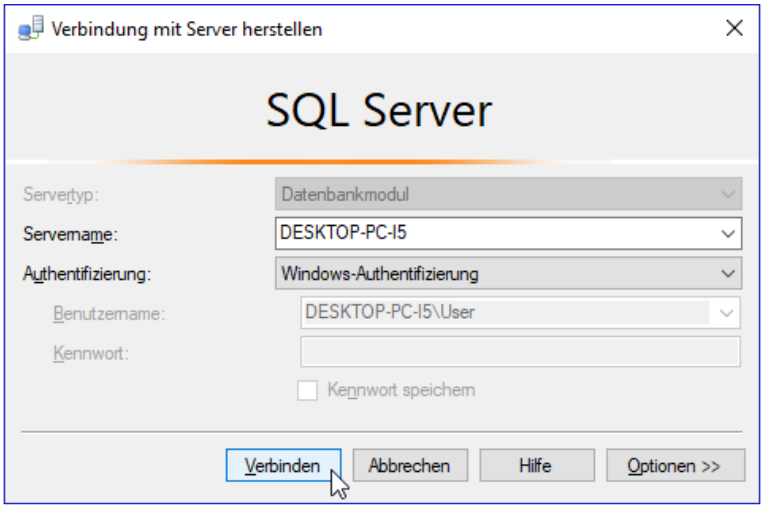


Bild 2: Bei der Windows-Authentifizierung sind die Felder Benutzername und Kennwort ausgegraut.

SQL Server: Zugriffe untersuchen mit XEvents

Wenn es darum ging, die Zugriffe auf eine SQL Server-Datenbank zu tracken – sei es zur Performance-Messung oder zur Optimierung einer Datenbank – war bis vor kurzem der SQL Server Profiler die Anwendung der Wahl. Mittlerweile gibt es allerdings einen in das SQL Server Management Studio integrierten Objekttyp namens »Erweiterte Ereignisse«. In diesem Beitrag wollen wir Ihnen diesen Objekttyp vorstellen. Das einfache Beispiel zu diesem Zweck stammt aus einem anderen Beitrag namens »SQL Server-Zugriff ohne gespeichertes Kennwort«. Hier wollen wir herausfinden, unter welchem Benutzer ein Zugriff auf eine SQL Server-Datenbank von Access aus erfolgt, bei dem offensichtlich gar kein Benutzer angemeldet ist.

Die **Erweiterten Ereignisse**, die wir in der deutschen Version des SQL Server Management Studio finden, **h**eißen im Englischen **Extended Events** oder – etwas stylischer – **XEvents**. Wir wollen daher in diesem Beitrag auch diese Bezeichnung wählen.

Die XEvents sind Ereignisse, die durch verschiedene Aktionen des SQL Servers ausgelöst werden und in die Sie eingreifen können, indem Sie verschiedene Informationen, die bei diesen Ereignissen anfallen, ausgeben oder speichern lassen.

Bis vor einigen SQL Server-Versionen war der SQL Server Profiler das Mittel der Wahl, wenn es um die Aufzeichnung und die Analyse von Ereignissen im SQL Server ging. Mit dem SQL Server 2008 wurden jedoch die XEvents eingeführt, die zunächst nur per T-SQL definiert werden konnten (ein Vorteil gegenüber den Ereignissen im Profiler), mittlerweile aber auch über die Benutzeroberfläche des SQL Server Management Studios angelegt werden können.

Access und XEvents

Für uns als Access-Entwickler werden die XEvents interessant, wenn wir eine Kombination aus Access-Frontend und SQL Server-Backend verwenden und hier den Datenverkehr zwischen dem Frontend und dem Backend analysieren wollen – sei es, um die Performance zu optimieren, die Menge der übertragenen Daten zu prüfen oder auch um andere Dinge herauszufinden – wie eben um das Problem zu lösen, dass uns im Beitrag **SQL Server-Zugriff ohne gespeichertes Kennwort** über den Weg gelaufen ist. Dort haben wir gerätselt, wie es sein kann, dass man mit einer Access-Anwendung auf die Tabellen einer SQL Server-Datenbank zugreifen kann, obwohl sich der geplante Benutzer überhaupt nicht angemeldet hat.

Was erfassen die XEvents?

Wenn wir hier alle Daten beschreiben wollten, die XEvents erfassen können, würde das den Rahmen sprengen. Uns interessieren aber auch nur diejenigen Ereignisse, die beim Zugriff unserer Access-Anwendung auf das SQL Server-Backend ausgelöst werden.

Um das auf zunächst in Objektform zu fassen: Wir können Zugriffe auf die per ODBC verknüpften Tabellen erfassen, Zugriffe über Aktionsabfragen und Aufrufe von SQL Server-Objekten wie gespeicherte Prozeduren, Funktionen und Trigger.

Wo erfassen die XEvents die Ereignisse?

Die mit XEvents erfassten Ereignisse können Sie beispielsweise in einer Datei speichern. Für unsere Zwecke gestaltete es sich allerdings als praktisch, die mit den Ereignissen erfassten Daten direkt im SQL Server Management Studio auszugeben.

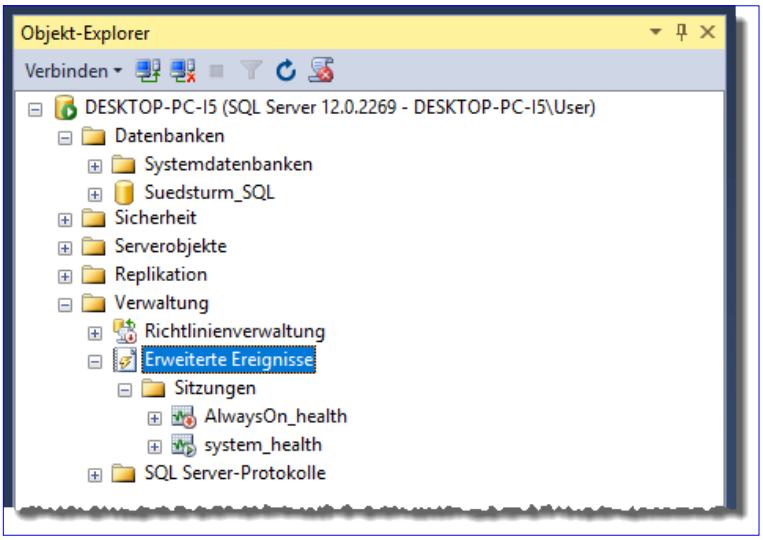


Bild 1: Die XEvents im Objekt-Explorer

XEvents im SQL Server Management Studio

In Visual Studio finden Sie die XEvents im Objekt-Explorer unter dem Eintrag **Verwaltung|Erweiterte Ereignisse** des Datenbankservers (siehe Bild 1).

Darunter befindet sich ein Eintrag namens **Sitzungen**. Hier finden Sie bereits zwei Einträge: **AlwaysOn_health** und **system_health**. **AlwaysOn_health** wird in Zusammenhang mit der Ausfallsicherheit des SQL Servers verwendet.

Interessant ist letzterer Eintrag: **system_health** speichert alle möglichen Informationen über den Betrieb des SQL Servers.

Diese können beispielsweise von Microsoft beim Auftreten von Problemen zur Analyse herangezogen werden.

Eine Sitzung ist, wenn Sie zuvor bereits mit dem SQL Server Profiler gearbeitet haben, mit einer Ablaufverfolgung zu vergleichen.

Was wir untersuchen wollen

Wie weiter oben erwähnt, gab es bei einer Frontend-Datenbank das Problem, dass man sich vermeintlich ohne vorherige Anmeldung an eine Datenbank im SQL Server anmelden konnte. Wir wollen uns ansehen, welche Informationen beim Zugriff auf die SQL Server-Datenbank ausgetauscht werden, um herauszufinden, wo der Denkfehler liegt.

Neue XEvents-Sitzung anlegen

Damit Sie besser verstehen, wie eine XEvents-Sitzung funktioniert, legen wir am besten einfach einmal eine an. Dazu klicken Sie mit der rechten Maustaste auf das Element **Sitzungen** im Objekt-Explorer und wählen aus

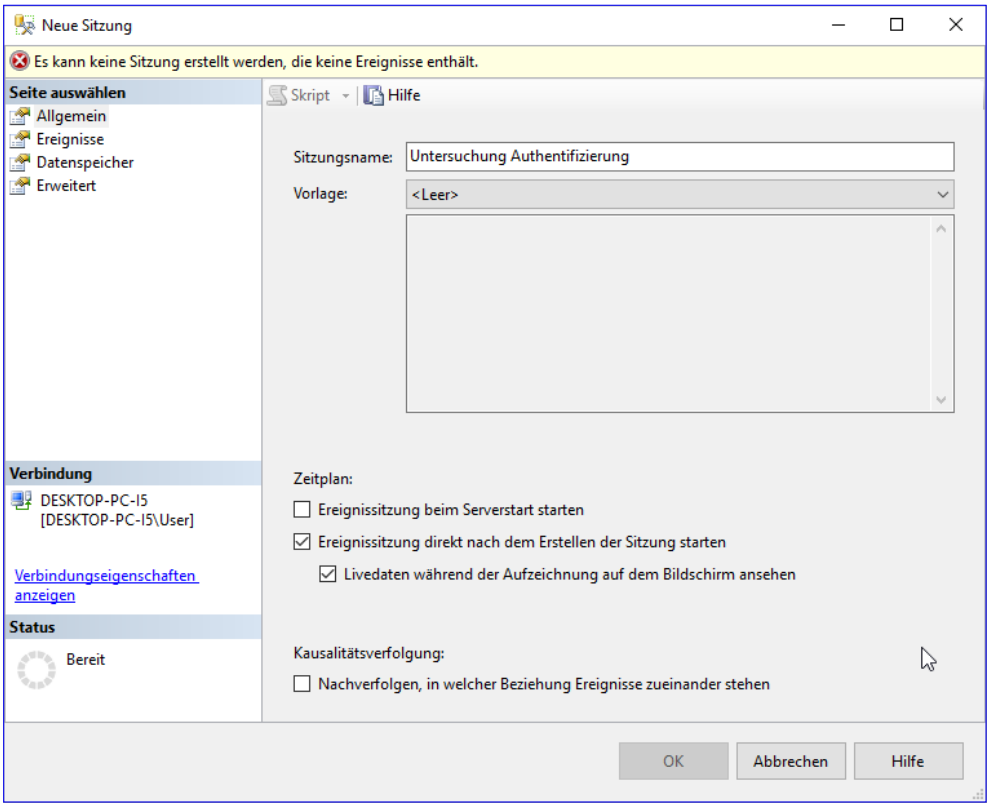


Bild 2: Anlegen einer neuen XEvents-Sitzung

dem Kontextmenü den Eintrag **Neue Sitzung...** aus. Es erscheint der Dialog **Neue Sitzung**. Hier gibt es, wie bei den meisten Dialogen zum Anlegen oder Bearbeiten von SQL Server-Elementen, links die Seitenauswahl und rechts die Inhalte der jeweiligen Seite.

Auf der ersten Seite namens **Allgemein** geben wir den Namen der Sitzung an, hier **Untersuchung Authentifizierung**. Das Feld **Vorlage** lassen wir auf **<Leer>** stehen. Außerdem wollen wir gleich loslegen, wenn wir die Sitzung angelegt haben, und stellen daher die Eigenschaft **Ereignissitzung direkt nach dem Erstellen der Sitzung starten** ein. Außerdem wollen wir **Livedaten während der Aufzeichnung auf dem Bildschirm ansehen** aktivieren (siehe Bild 2). Wir zeigen Ihnen später, wie Sie diese Aktionen später manuell starten.

Ereignisse definieren

Damit wechseln wir auf die zweite Seite namens **Ereignisse**. Auf dieser Seite finden Sie links ein Listenfeld mit allen verfügbaren Ereignissen – sehr vielen Ereignissen. Wir benötigen nur sehr wenige davon. Damit wir diese finden, geben wir einen Teil des Namens in das Feld mit dem Inhalt **Ereignisse suchen** ein, zum Beispiel **completed**. Damit werden nur noch die in Bild 3 sichtbaren Ereignisse angezeigt. Klicken wir auf eines der Ereignisse, zeigt das Feld links unter der Liste die Beschreibung des Ereignisses an. Die Liste rechts unter der Liste liefert alle

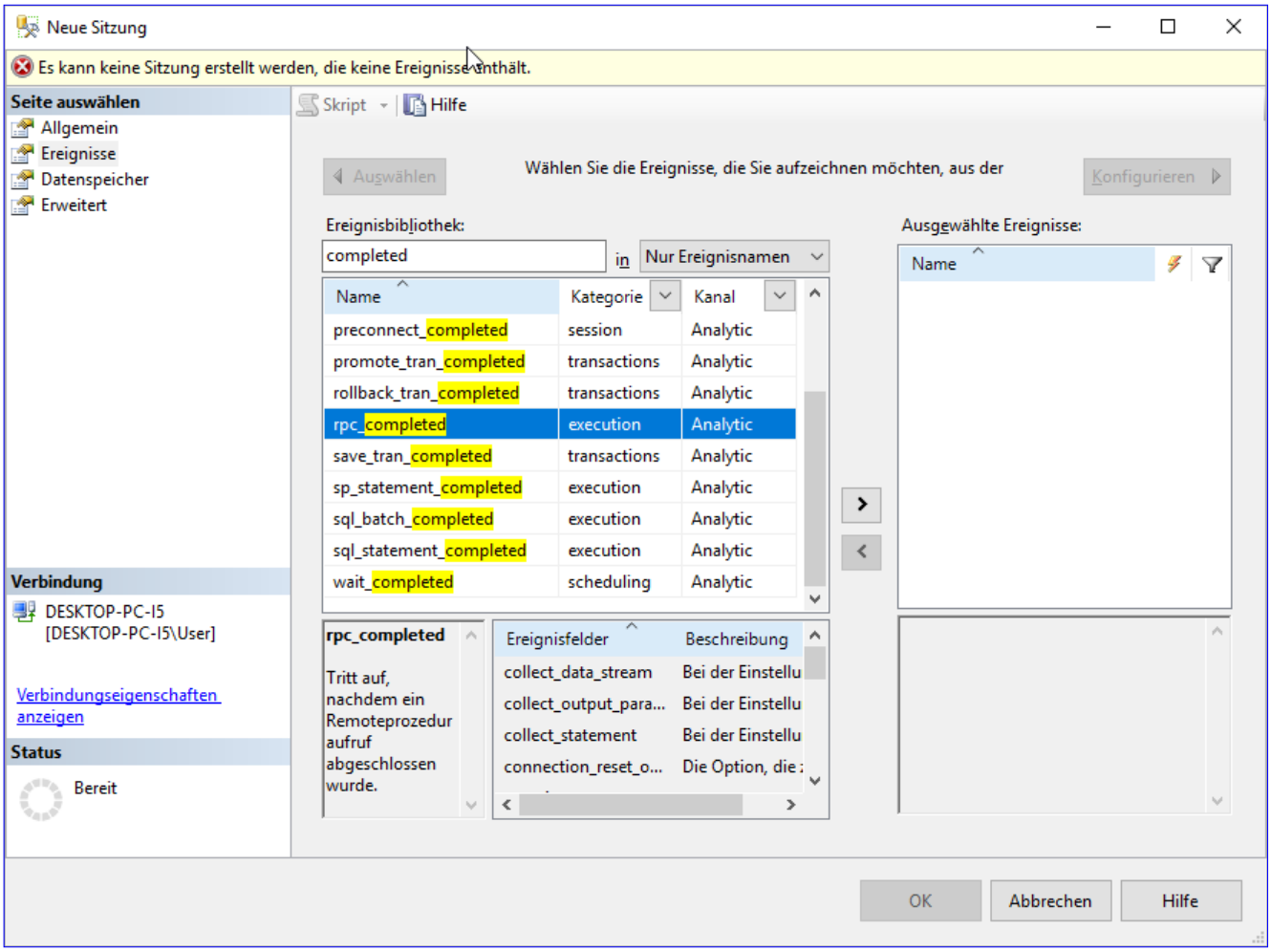


Bild 3: Suchen nach den benötigten Ereignissen