HAUFE.
Ausgabe 05/2023

ACCESS
IM UNTERNEHMEN

TURBO FÜR ACCESS 32-BIT

Nutzen Sie in Access in der 32-Bit-Version genauso viel Speicher wie in der 64-Bit-Version (ab Seite 49)

In diesem Heft:

MEMOFELDER MIT SEHR LANGEN TEXTEN

Auch Memofelder haben Grenzen. In diesem Beitrag lernen wir diese kennen und holen mehr heraus.

SEITE 2

SQL SERVER-BACKUP VERSCHLÜSSELN

Sensible Daten sollten auch als Backup gut geschützt sein. Eine passende Anleitung liefert dieser Beitrag.

SEITE 36

PERFORMANCE UND SPEICHER OPTIMIEREN

Durch Dekompilieren einer Datenbank werden alte und kaputte Elemente entsorgt. Wir zeigen einen komfortablen Weg.

SEITE 65

Höher, schneller, weiter

Microsoft Access ist nicht gerade das Mitglied der Office-Produktlinie, das mit den meisten Neuerungen ausgestattet wird. Das betrifft auch den Bereich Performance. Manchmal hakt es bei Datenbankanwendungen, was verschiedene Gründe haben kann. Und wenn der Hersteller schon keine Abhilfe schafft, müssen wir Entwickler uns eben selbst helfen. In dieser Ausgabe schauen wir uns gleich in drei Beiträgen an, wie das möglich ist.

Wer mit der 32-Bit-Version von Access arbeitet, ist Access sicher schon einmal beim Öffnen von Formularen oder auch beim Ausführen von VBA-Code mit Fehlermeldungen ausgestiegen, die auf mangelnde Ressourcen hinweisen. Wer dann im Task Manager von Windows nach Nadelöhren fahndet, sucht an der falschen Stelle. Mehr Erfolg haben wir mit einem Tool namens VMMap, das kostenlos von Microsoft angeboten wird. Dieses liefert uns Einblicke in den für Microsoft Access verfügbaren virtuellen Speicher. Unter Access in der 32-Bit-Version ist dieser rund zwei Gigabyte groß, unter der 64-Bit-Version rund vier Gigabyte. Zieht man ungefähr ein Gigabyte ab, das bereits von Access selbst verwendet wird, ohne dass wir irgendwelche Objekte öffnen oder Code ausführen, verfügt die 64-Bit-Version also etwa über den dreifachen virtuellen Speicher. Es gibt allerdings einen Trick, mit dem man diesen Speicher auch für die 32-Bit-Version verfügbar machen kann. Wie das gelingt, zeigen wir im Beitrag **Schneller, weiter, höher mit LAA für Access 32-Bit** ab Seite 49.

Wenn Sie einmal untersuchen wollen, wie sich die verschiedenen Elemente Ihrer Datenbankanwendung auf die verfügbaren Ressourcen auswirken, finden Sie im Beitrag **Access-Speicher überwachen mit VMMap** ab Seite 58 weitere Informationen zu dem oben bereits erwähnten Tool **VMMap**. Hier zeigen wir, wie Sie das Tool nutzen und auch, welche Beobachtungen wir bei verschiedenen Konstellationen hinsichtlich der verfügbaren Ressourcen gemacht haben.

Eine weitere Stellschraube für das Freisetzen von Performance und Speicherplatz (diesmal auf der Festplatte) ist das Dekompilieren von Access-Datenbanken. Dabei werden nicht nur alte, ungenutzte Code-Fragmente entfernt, sondern dies reduziert gegebenenfalls auch die Größe der Datenbankdatei massiv. Noch einen besseren Effekt erhält man, wenn man das Dekompilieren einer Datenbank mit dem Komprimieren und Reparieren kombiniert. Alles Wichtige zu diesem Thema beschreiben wir im Beitrag **Dekompilieren leicht gemacht** ab Seite 65.

Manchmal hakt es, wenn man Memofeldern zu lange Texte zuweisen möchte. Wie wir den Platz eines Memofeldes zu 100% nutzen können, zeigen wir im Beitrag **Memofelder mit sehr langen Texten** ab Seite 2.

Rund um den SQL Server haben wir diesmal gleich zwei Beiträge. Bernd Jungbluth zeigt in **SQL Server-Security – Teil 8: Datenbankrollen** ab Seite 6, wie wir Datenbankrollen für die Sicherheit in SQL Server-Datenbanken mit Access-Frontends nutzen können. Und unter dem Titel **SQL Server: Verschlüsselte Backups erstellen** zeigen wir ab Seite 36, wir man verhindert, dass sensible Daten in unverschlüsselten Sicherungen landen.

Schließlich schauen wir uns im Beitrag **Outlook: E-Mail-Absender per VBA einstellen** ab Seite 70 noch an, wie wir den Absender einer Outlook-E-Mail einstellen können. Und nein: Das gelingt nicht durch einfaches Zuweisen der entsprechenden Adresse an eine Eigenschaft.

Nun aber viel Spaß beim Stöbern in der neuen Ausgabe!

Ihr André Minhorst

Memofelder mit sehr langen Texten

In Feldern, deren Felddatentyp früher Memofeld hieß und heute »langer Text« genannt wird, kann man bis zu einem Gigabyte an Daten speichern. Allerdings gibt es verschiedene Einschränkungen bezüglich der Datenmenge. So kann man je nach der Speichermethode nur sehr viel weniger Zeichen eingeben. Und erst recht kann man nicht den kompletten Inhalt eines Memofeldes in einem Textfeld anzeigen, wenn dieses mehr als eine bestimmte Menge Zeichen enthält. In diesem Artikel schauen wir uns einmal an, welche Einschränkungen es gibt, wie man diese gegebenenfalls umgehen kann und welchen Nutzen Memofelder überhaupt haben, wenn man mehr als den anzeigbaren Text eingibt.

Wozu Memofelder mit umfangreichen Inhalten?

Bevor wir uns darum kümmern, welche Besonderheiten Memofelder mit sehr umfangreichen Inhalten überhaupt aufweisen, schauen wir uns an, wozu wir diese überhaupt nutzen können.

Dazu gehören die Folgenden:

- Kommentarfunktion: Wenn Sie eine Kommentarfunktion in Ihre Access-Datenbank integrieren möchten, können Sie Memofelder verwenden, um lange Texte von Benutzern zu speichern.
- Dokumentation: Wenn Sie eine Datenbank für ein Projekt erstellen, kann ein Memofeld verwendet werden, um detaillierte Beschreibungen oder Anweisungen zu speichern, um das Projekt besser zu dokumentieren.
- Notizen: Wenn Sie Ihre persönlichen Notizen in Access speichern möchten, können Sie ein Memofeld verwenden, um Ihre Gedanken und Ideen zu speichern.
- Fehlerprotokollierung: Wenn Sie eine Fehlerprotokollierung in Ihre Access-Anwendung einbauen möchten, können Sie Memofelder verwenden, um die Beschreibungen von Fehlern zu speichern, die bei der Verwendung ausgelöst werden.
- Beschreibungen: Wenn Sie eine Datenbank für den Verkauf von Produkten erstellen, können Sie Memofelder verwenden, um detaillierte Beschreibungen der Produkte zu speichern.

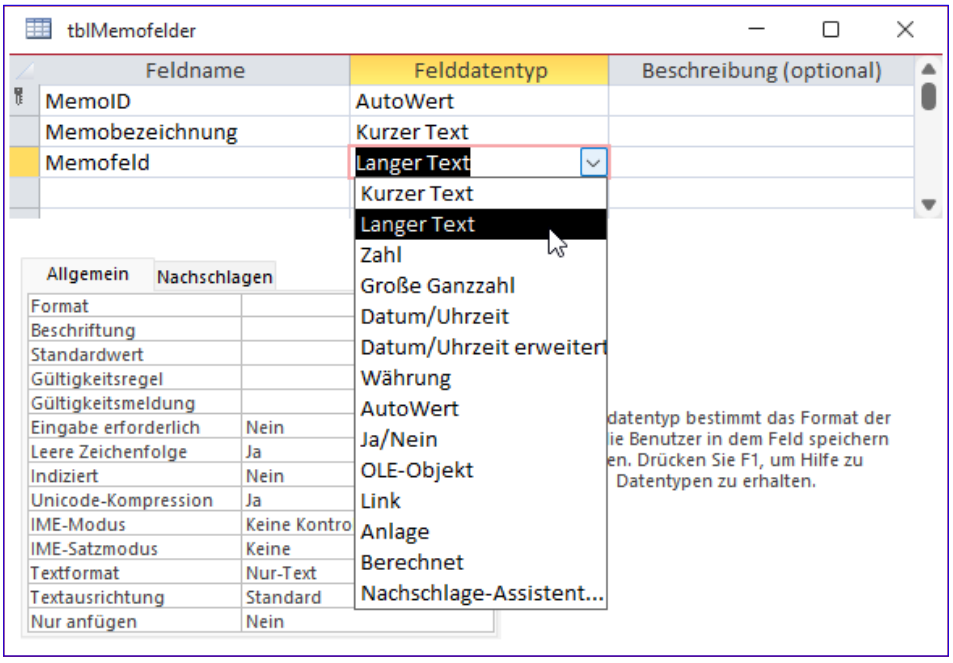

Bild 1: Tabelle mit einem neuen Memofeld

Memofeld anlegen

Bevor wir mit einigen Experimenten starten, erstellen wir eine Tabelle mit einem Feld mit dem Datentyp **Memofeld** beziehungsweise **Langer Text**. Dazu fügen wir einer Tabelle in der Entwurfsansicht einfach ein neues Feld hinzu, in diesem Fall namens **Memofeld**, und stellen seinen Datentyp auf **Langer Text** ein (siehe Bild 1).

Formular an ein Memofeld binden

Anschließend erstellen wir ein neues Formular und stellen seine Eigenschaft **Datensatzquelle** auf die Tabelle **tblMemofelder** ein. Dann ziehen wir alle Felder der Datensatzquelle aus der Feldliste in den Entwurf des Formulars. Für das Memofeld stellen wir die Eigenschaften **Horizontaler Anker** und **Vertikaler Anker** jeweils auf **Beide** ein, damit es sich mit dem Formular vergrößert (siehe Bild 2).

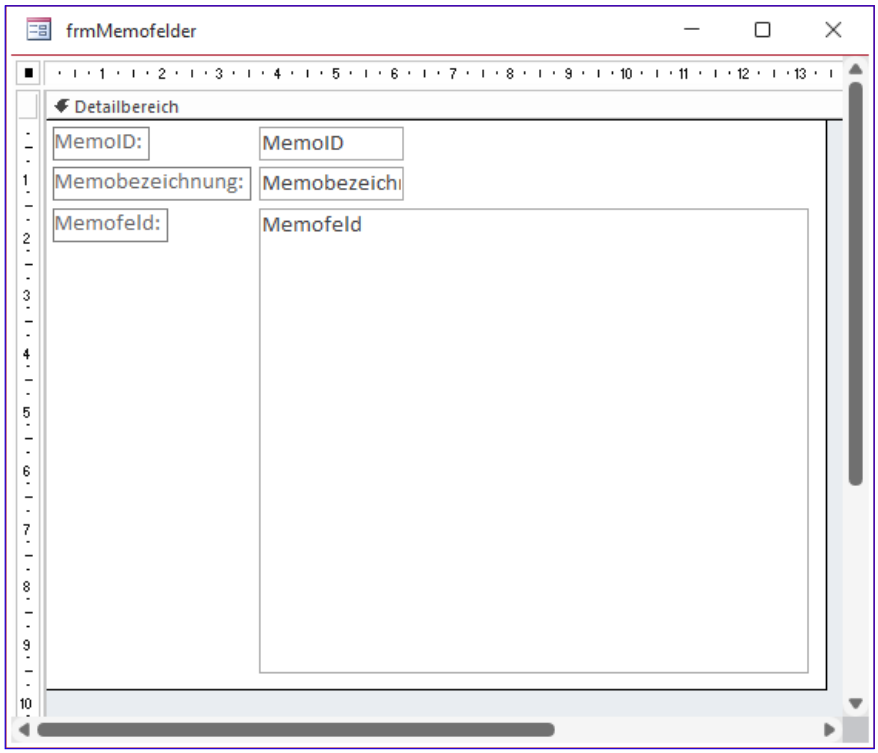

Bild 2: An eine Tabelle mit einem Memofeld gebundenes Formular

Experimente mit dem Memofeld

Damit starten wir in eine Reihe von Experimenten. Dabei wollen wir uns Folgendes ansehen:

- Wie viele Zeichen kann man direkt in ein Memofeld einfügen?
- Wie viele Zeichen können wir dem Textfeld per Code zuweisen?
- Wie viele Zeichen kann man über ein Textfeld in ein Memofeld einfügen?
- Wie viele Zeichen lassen sich per Execute in ein Memofeld einfügen?
- Wie viele Zeichen lassen sich per Recordset in ein Memofeld einfügen?
- Wie viele Zeichen eines Memofeldes zeigt ein Textfeld maximal an?

Starten wir mit den Experimenten!

Experimente: Wie viele Zeichen kann man je Methode in ein Memofeld einfügen?

Wir haben die Idee, dass es sich durchaus um einige tausend Zeichen handeln kann. Wie aber können wir eine entsprechende Zeichenmenge generieren? Dazu können wir die **String**-Funktion nutzen. Sie erwartet zwei Parameter: die Anzahl der zu erzeugenden Zeichen und das Zeichen, das verwendet werden soll. Das funktioniert beispielsweise so:

```
? String(10, "A")
AAAAAAAAAA
```

Die Frage ist nur: Wie lange Zeichenketten können wir mit dieser Funktion generieren und reichen diese für unsere Zwecke aus? Auf unserem Rechner konnten wir mit der

SQL Server-Security – Teil 8: Datenbankrollen

Bernd Jungbluth, Horn (info@berndjungbluth.de)

Berechtigungskonzepte sind mit einer Vielzahl von Fachbegriffen verbunden. Separation of Duties, Principle of Least Privilege, Role Based Access Control, Discretionary Access Control etc. pp. Sie behandeln die Vergabe von Zugriffsrechten auf unterschiedlichen Sicherheitsniveaus. Im Kern jedoch dient jede dieser Methoden der Vertraulichkeit und Integrität der Daten. Die Anwender dürfen nur auf die Ressourcen zugreifen, die sie zum Erfüllen ihrer Aufgaben benötigen. Dabei gilt es Überschneidungen von Zuständigkeiten und somit den Missbrauch von Daten zu vermeiden. SQL Server unterstützt zur Umsetzung dieser Anforderungen gleich mehrere Konzepte.

In den bisherigen Beiträgen dieser Reihe haben Sie einige Varianten möglicher Berechtigungskonzepte kennengelernt. Das aktuelle Berechtigungskonzept der Beispielumgebung basiert auf der Windows-Authentifizierung und drei Datenbankrollen. Die Anwender melden sich mit ihren Windows-Benutzerkonten am SQL Server an. Wobei im SQL Server als Anmeldungen nicht die Benutzerkonten der Anwender hinterlegt sind, sondern die der Windows-Gruppen **Personal** und **Vertrieb**. So authentifiziert sich Frau Bienlein als Mitglied der Windows-Gruppe **Vertrieb** über die gleichnamige Anmeldung am SQL Server. Ähnlich ist es beim Anwender **Stromberg**. Er ist Mitglied der Windows-Gruppe **Personal** und für diese gibt es im SQL Server die Anmeldung **Personal**.

Der eigentliche Datenzugriff ist in der Datenbank geregelt. Zu jeder Anmeldung existiert dort ein entsprechender Benutzer. Die beiden Benutzer **Personal** und **Vertrieb** sind Mitglieder der Datenbankrollen **db_datareader**, **db_datawriter** und **edb_execute**. Durch diese Zuordnung dürfen die Benutzer und somit letztendlich die Mitglieder der Windows-Gruppen die Daten aller Tabellen lesen, ändern, ergänzen und löschen sowie alle gespeicherten Prozeduren ausführen. Lediglich den Vertriebsmitarbeitern ist der Zugriff auf die Daten der Personalabteilung nicht gestattet. Das wird mit einer erweiterten Rechtevergabe sichergestellt. Diese verweigert dem Benutzer **Vertrieb** den Zugriff auf die Tabellen **Bewerber**, **Mitarbeiter** und **Stellen** sowie der gespeicherten Prozedur **pSelectGeburtstagsliste**.

Dieses Berechtigungskonzept bietet bereits ein recht hohes Sicherheitsniveau. Dennoch ist es nicht detailliert genug. Die Anwender der Windows-Gruppe **Vertrieb** haben zwar keinen Zugriff auf die Personaldaten, auf alle anderen Daten können sie jedoch zugreifen. Um diese Sicherheitslücke zu schließen, bedarf es einer dedizierten Vergabe der Zugriffsrechte. Ein solch detailliertes Berechtigungskonzept benötigt man, wenn die Datenbank Informationen für verschiedene Zwecke speichert. Die Beispieldatenbank **WaWi_SQL** enthält neben den Daten des Vertriebs und der Personalabteilung die Daten des Einkaufs. Die Zugriffsrechte zu diesen Daten sollen aus Gründen der Datensicherheit und des Datenschutzes getrennt werden.

Datensicherheit und Datenschutz

Wie wichtig eine Trennung aus Gründen der Datensicherheit ist, haben Sie in den letzten Beiträgen anhand der Aktionen von Frau Bienlein gesehen. Aktuell hat sie Zugriff auf die Daten des Einkaufs und kann Lieferanten mitsamt Eingangsrechnungen und Wareneingängen anlegen. Der

monatliche Zahlungslauf der Buchhaltung überweist dann die Beträge der nicht realen Eingangsrechnungen an das Bankkonto des fiktiven Lieferanten, also an Frau Bienlein. Das neue Berechtigungskonzept soll Datenmissbrauch dieser Art verhindern.

Eine detaillierte Rechtevergabe ist ebenso aus Gründen des Datenschutzes erforderlich. Aktuell haben die Mitarbeiter der Personalabteilung vollen Zugriff auf alle Daten der Datenbank. Hierunter fallen die Daten der Kunden und deren Ansprechpartner. Dieser Datenzugriff ist nicht datenschutzkonform. Die Daten der Kunden und Ansprechpartner werden zum Zweck der Vertragserfüllung und Kundenbindung in der Datenbank gespeichert. Ein Zugriff der Personalabteilung lässt sich mit diesem Zweck nicht vereinbaren.

Somit wird die Zweckbindung als einer der Grundsätze zur Verarbeitung von personenbezogenen Daten mit dem aktuellen Berechtigungskonzept nicht erfüllt.

Datensicherheit und Datenschutz sind zwei Aspekte zur Bewertung des Sicherheitsniveaus. Beide fordern die Vertraulichkeit und Integrität der Daten. Aber was genau verbirgt sich hinter dieser Forderung?

Vertraulichkeit und Integrität der Daten

Das Bundesamt für Sicherheit in der Informationstechnik (BSI) betrachtet die Vertraulichkeit und Integrität der Daten zusammen mit deren Verfügbarkeit als die „Schutzziele oder auch Grundwerte der Informationssicherheit“.

Vertraulichkeit beschreibt das BSI als den „Schutz vor unbefugter Preisgabe von Informationen“ oder etwas konkreter mit „Vertrauliche Daten und Informationen dürfen ausschließlich Befugten in der zulässigen Weise zugänglich sein.“.

Integrität „bezeichnet die Sicherstellung der Korrektheit (Unversehrtheit) von Daten und der korrekten Funktionsweise von Systemen“. Daten verlieren ihre Integrität, wenn „diese unerlaubt verändert“ wurden.

Die hier aufgeführten Definitionen stammen aus dem Glossar des IT Grundschutz Kompendiums. Ein relevantes Werk zum Thema IT-Sicherheit.

Eine Maßnahme zur Gewährleistung der Vertraulichkeit und Integrität der Daten ist ein Berechtigungskonzept. Wobei dessen Detaillierungsgrad von Ihren Sicherheitsanforderungen abhängt. Der Aufwand in einem Bioladen wird geringer sein als in einem Krankenhaus. Das Sicherheitsniveau Ihres Unternehmens stellt also die Grundlage des Berechtigungskonzepts dar. Zur Umsetzung beginnt man mit einer Bestandsaufnahme.

Ein detailliertes Berechtigungskonzept

Ziel der Bestandsaufnahme ist eine Aufstellung der im Unternehmen anfallenden Aufgaben sowie deren Aufgabenträger. Sie müssen also herausfinden, wer im Unternehmen welche Aufgaben erledigt. Die ersten Fragen lauten daher: Wer macht hier eigentlich was? Und warum? Auf der Suche nach der Antwort erleben Sie möglicherweise die eine oder andere Überraschung.

Sie werden feststellen, dass Personen Aufgaben erledigen, von denen Sie das nicht erwartet hätten. Und Sie werden sehen, dass manche Aufgabe doppelt und dreifach von Personen in unterschiedlichen Abteilungen bearbeitet wird. Wohingegen sich um einige Aufgaben niemand kümmert. Sie werden erleben, wie sich Personen an Aufgaben klammern und wie andere wiederum die Aufgaben so schnell wie möglich loswerden möchten. Die menschliche Komponente beim Erstellen eines Berechtigungskonzepts sollten Sie nicht unterschätzen.

Das klingt jetzt nicht so erfreulich für Sie? Dann betrachten Sie es mal von einer anderen Seite. Eine Übersicht der Tätigkeiten Ihrer Kollegen bietet einige Vorteile. Vielleicht erkennen Sie Datenzugriffe, die es aus Gründen der Datensicherheit oder des Datenschutzes gar nicht geben dürfte. Möglicherweise entdecken Sie einiges an Verbesserungspotential in der Aufgabenverteilung. Eine Neustrukturierung der Aufgaben wiederum kann Ihnen

Einsparmöglichkeiten aufzeigen. Sie brauchen vielleicht nicht auf allen Rechnern zwingend eine Access-Lizenz und eventuell ist eine Lizenzierung des SQL Servers nach Client Access Licence günstiger als eine Lizenzierung pro Prozessor.

Bei der Bestandsaufnahme sollten Sie neben den menschlichen Aufgabenträgern ebenso die maschinellen Aufgabenträger berücksichtigen. Im Hinblick auf ein Berechtigungskonzept für Datenbanken spielen hierbei die Applikationen eine wichtige Rolle. Welche Applikationen werden zur Bewältigung welcher Aufgaben eingesetzt? Welche Datenbanken gehören zu den Applikationen? Und natürlich stellt sich letztendlich die Frage, welche Anwender mit diesen Applikationen arbeiten.

Was die Anwender betrifft, ist für Sie als Datenbankadministrator bereits ein Teil der Arbeit getan. Deren Windows-Benutzerkonten sind in den meisten Fällen in Windows-Gruppen zusammengefasst. Der IT-Systemadministrator hat sich bei der Gruppenbildung bereits Gedanken über die Gruppenzugehörigkeit der Anwender gemacht. Nicht selten werden in diesen Gruppen die Abteilungen des Unternehmens abgebildet. Eine weitere Art der Gruppenbildung basiert auf Arbeitsabläufen und Prozessen, insbesondere wenn hierbei mehrere Abteilungen involviert sind.

Des Weiteren gibt es Gruppen zur Funktionstrennung, um Interessenskonflikte beziehungsweise Überschneidungen von Zuständigkeiten zu vermeiden. Wie auch immer, Sie als Datenbankadministrator sollten zunächst prüfen, ob Sie die Windows-Gruppen ohne weiteres übernehmen können. Sind die Gruppierungen nicht detailliert genug, erstellen Sie zusammen mit Ihrem IT-Systemadministrator weitere Gruppen.

Im SQL Server legen Sie zu den einzelnen Windows-Gruppen die Anmeldungen an und ordnen diese der Datenbank zu. Sie kennen diese Vorgehensweise bereits. Durch die Zuordnungen werden in der Datenbank die Benutzer zu den Anmeldungen erstellt. Dort erfolgt dann durch die Mitgliedschaft der Benutzer in den einzelnen Datenbankrollen die eigentliche Berechtigungsvergabe.

Eine Datenbankrolle enthält die Zugriffsrechte an den Tabellen, die die Daten für die Aufgaben speichern und an der Geschäftslogik, die in den gespeicherten Prozeduren definiert ist. So bestimmen Sie letztendlich, an welchen Tabellen die Mitglieder der Datenbankrolle eines, mehrere oder gar alle der Rechte **SELECT**, **INSERT**, **UPDATE** und **DELETE** besitzen. Ebenso definieren Sie in der Datenbankrolle welche gespeicherten Prozeduren per **EXECUTE** ausgeführt werden dürfen. Wie Sie vom aktuellen Berechtigungskonzept her wissen, können Sie auf diese Art nicht nur Rechte vergeben, sondern auch explizit verweigern.

Dieses Rollenprinzip ist als Role Based Access Control (RBAC) bekannt. Natürlich ist eine Vergabe der Rechte ebenso direkt am Benutzer selbst möglich. Im aktuellen Berechtigungskonzept ist dies bei dem Benutzer **Vertrieb** der Fall. Ihm ist der Zugriff auf die Tabellen und gespeicherten Prozeduren der Personalabteilung explizit verweigert. Discretionary Access Control (DAC) lautet die Bezeichnung für diese direkte Art der Rechtevergabe. In den meisten Fällen ist sie jedoch nicht empfehlenswert. Angenommen, die Außendienstmitarbeiter sollen die gleichen Zugriffsrechte wie die Vertriebsmitarbeiter erhalten. Dazu legen Sie für die Windows-Gruppe **ADM** eine Anmeldung im SQL Server an und ordnen diese der Datenbank **WaWi_SQL** zu. Bei einer Rechtevergabe pro Benutzer müssten Sie die Rechte des Benutzers **ADM** an die des Benutzers **Vertrieb** anpassen. Jede Änderung der Zugriffsrechte wäre dann bei beiden Benutzern zu pflegen. Ist die Rechtevergabe jedoch in einer Datenbankrolle definiert, nehmen Sie zusätzlich zum Benutzer **Vertrieb** den neuen Benutzer **ADM** in die Datenbankrolle auf. Weitere Änderungen der Zugriffsrechte finden ausschließlich in der Datenbankrolle statt und stehen direkt für die dort zugeordneten Benutzer zur Verfügung, im Beispiel für die Vertriebsmitarbeiter und die Außendienstmitarbeiter. Am besten denken Sie immer in Datenbankrollen. Das erleichtert Ihnen die Pflege der Rechtevergabe.

Apropos erleichtern. Der wohl wichtigste Leitspruch in der Security lautet KISS – Keep It Simple, Stupid. Halten Sie Ihr Berechtigungskonzept so einfach und übersichtlich wie möglich. Je komplizierter es ist, desto eher wird im Falle eines fehlenden Zugriffsrechts die Rechtevergabe schnell mal erweitert. Hauptsache, der Anwender kann arbeiten. Die Nachjustierung im Sinne einer Korrektur der Zugriffsrechte erfolgt aus Zeitmangel eher selten. Wodurch die komplette Arbeit zur Rechtevergabe für die Katz ist.

Die Zugriffsberechtigungen nach Role Based Access Control (RBAC) und Discretionary Access Control (DAC) beziehen sich auf die Datenbankobjekte. Zugriffsrechte auf einzelne Datensätze sind hier nicht enthalten. Diese Art der Rechtevergabe fällt unter den Begriff Mandatory Access Control (MAC). Hier basieren die Zugriffsrechte auf Regeln. Regeln, die beispielsweise den Vertriebsmitarbeitern den Zugriff auf die Kundendaten ihrer Zuständigkeiten begrenzen. So kann ein Mitarbeiter nur seine Kunden sehen und nicht die seiner Kollegen. In SQL Server lässt sich ein derartiger Datenzugriff mittels Row Level Security (RLS) realisieren.

Der Werkzeugkasten

Das Berechtigungskonzept zur Datenbank **WaWi_SQL** soll dem Rollenprinzip entsprechen. Dazu fassen Sie einzelne Datenbankobjekte in Datenbankrollen zusammen und legen dort deren Zugriffsrechte fest. Hierbei orientieren Sie sich an den Aufgaben. Welche Datenbankobjekte werden zum Erfüllen der Aufgaben benötigt? Welche Rechte an den einzelnen Datenbankobjekten sind dazu erforderlich? Bildlich gesprochen nehmen Sie mit den Datenbankobjekten das benötigte Werkzeug und legen es in den Werkzeugkasten, in die Datenbankrolle. Welche Person letztendlich den Werkzeugkasten in die Hand nimmt und sich an die Aufgaben macht, ist für das Zusammenstellen des Werkzeugs irrelevant.

Wie kommt der Werkzeugkasten zur Person? Durch die Zuordnung des Benutzers zur Datenbankrolle. Wobei der Benutzer hier eher einer Stellenbeschreibung ähnelt. Wer diese Stelle besetzt, ergibt sich durch die Verknüpfung des Benutzers zur Anmeldung, welche wiederum auf eine Windows-Gruppe verweist. Somit haben alle Mitglieder der Windows-Gruppe Zugriff auf den Werkzeugkasten, sprich auf die Ressourcen, die sie zur Bewältigung ihrer Aufgaben benötigen. Nicht mehr und nicht weniger. Diese Einschränkung wird als Principle Of Least Privilege (PoLP) bezeichnet.

An dieser Stelle zeigt sich wieder einmal die Stärke der mehrstufigen Sicherheitsarchitektur des SQL Servers. Die zur Erfüllung der Aufgaben erforderlichen Zugriffsrechte sind innerhalb der Datenbank definiert. Während die Anmeldungen der für die Aufgaben zuständigen Personen außerhalb der Datenbanken verwaltet werden. Verweisen diese Anmeldungen auf Windows-Gruppen, findet deren Definition und Verwaltung nicht einmal im SQL Server statt. Eine klare Trennung von Aufgaben und Personen.

Eine solche Trennung bietet einige Vorteile, zum Beispiel im Fall einer Urlaubsvertretung. Für die Rechtevergabe reicht es aus, dass der IT-Systemadministrator das Windows-Benutzerkonto der Urlaubsvertretung in die entsprechende Windows-Gruppe aufnimmt. Dadurch erhält der Anwender alle erforderlichen Zugriffsrechte an den Ressourcen, die er zum Bewältigen der vorübergehenden Aufgaben benötigt, inklusive der Rechte für die Datenzugriffe in den Datenbanken. Ein Nachjustieren der Zugriffsrechte innerhalb der Datenbanken ist in der Regel nicht erforderlich.

Der wohl wichtigste Aspekt bei der Zuordnung der Personen zu den Aufgaben ist das Prinzip der Funktionstrennung. Es gilt, Interessenskonflikte sowie die Möglichkeit krimineller Handlungen zu vermeiden. Dieses als Separation of Duties (SoD) oder Segregation of Duties bekannte Prinzip fordert, dass mehrere zusammenhängende Teilaufgaben nicht von ein und derselben Person durchgeführt werden. So sollte ein und dieselbe Person nicht für die Umsetzung einer Aufgabe und gleichzeitig für deren Abschluss beziehungsweise Kontrolle verantwortlich

sein. Zum Beispiel für das Anlegen von Lieferanten und das Erfassen von Bestellungen und Wareneingängen.

Nach dem Prinzip der Funktionstrennung wäre es somit in der Beispielumgebung erforderlich, die Zugriffsrechte zu den Daten der Lieferanten von denen der Bestellungen und Wareneingänge zu trennen. Ohne diese Trennung sind die Mitarbeiter wie zuletzt Frau Bienlein in der Lage, einen fiktiven Lieferanten mitsamt Bestellungen anzulegen und den zugehörigen Wareneingang zu buchen. Vermeiden Sie unbedingt derartige Überschneidungen von Zuständigkeiten und somit den Missbrauch von Daten. Verteilen Sie die Zugriffsrechte in mehrere Datenbankrollen und ordnen Sie diesen unterschiedlichen Benutzer zu.

Sie sehen, ein Berechtigungskonzept kann schnell vielschichtig werden. Hinzu kommt, dass in einer über Jahre gewachsenen Datenbank-Applikation die Definition der Datenbankrollen gar mal nicht so einfach ist. Sie stoßen mit hoher Wahrscheinlichkeit auf Datenbankobjekte, die für mehrere Aufgaben erforderlich sind. In diesen Fällen sind Sie bei der Rechtevergabe bitte nicht zu großzügig. Folgen Sie weiterhin dem Principle Of Least Privilege und erteilen Sie den Anwendern nur die Rechte, die sie für ihre tägliche Arbeit benötigen. Vergeben Sie zu viele Rechte, ermöglichen Sie unerlaubte Zugriffe und den Missbrauch der Daten. Vergeben Sie zu wenig Rechte, können die Mitarbeiter ihre Aufgaben nicht wahrnehmen. Die Folge ist eine Fehlermeldung und die Lösung zu diesem Fehler meist eine zu hohe Rechtevergabe wie die Zuordnung des Benutzers zur Datenbankrolle **db_owner** oder gar der Anmeldung zur Serverrolle **sysadmin**.

Bestandsaufnahme

Genug der Theorie und Fachbegriffe. Wie gehen Sie am besten an die Bestandsaufnahme heran? Insbesondere im Hinblick auf das Berechtigungskonzept einer Datenbank? Schauen Sie sich den Sinn und Zweck der Applikationen zur Datenbank an. Welche Aufgaben werden mit diesen Applikationen erfüllt? In der Beispielapplikation sind es die des Vertriebs, des Einkaufs, des Personalwesens und der Buchhaltung. Diese vier Abteilungen sind eine gute Basis für die Definition der Windows-Gruppen und der Datenbankrollen.

Jeder Mitarbeiter dieser Abteilungen besitzt ein eigenes Windows-Benutzerkonto. Diese sind in Windows-Gruppen zusammengefasst. Frau Bienlein gehört zur Windows-Gruppe **Vertrieb**, Herr Stromberg zur Windows-Gruppe **Personal**, Herr Eberhofer zur Windows-Gruppe **Einkauf** und Frau Kling zur Windows-Gruppe **Buchhaltung**.

Im SQL Server gibt es zu jeder dieser Windows-Gruppen eine Anmeldung. Die Anmeldungen sind der Datenbank **WaWi_SQL** zugeordnet, wodurch es dort zu jeder Anmeldung und somit indirekt zu jeder Windows-Gruppe einen Benutzer gibt. Das aktuelle Berechtigungskonzept basiert auf einer pauschalen Rechtevergabe per Systemdatenbankrollen und einer eigenen allgemeingültigen Datenbankrolle zur Ausführung von gespeicherten Prozeduren. Das werden Sie nun ändern. Die Datenzugriffe sollen ausschließlich über eigene Datenbankrollen mit dedizierten Rechten erfolgen.

Als ersten Schritt entfernen Sie die bestehende Rechtevergabe. Beginnen Sie mit dem Benutzer **Personal**. Öffnen Sie das SQL Server Management Studio mit den Rechten eines Datenbankadministrators und navigieren Sie im Objekt-Explorer über den Eintrag **Datenbanken\ WaWi_SQL\Sicherheit** zum Ordner **Benutzer**. Wie Sie in Bild 2. sehen, setzt sich der Benutzername aus der Bezeichnung der Domäne und der Windows-Gruppe zusammen. Wobei in Ihrer Umgebung natürlich die Bezeichnung Ihrer Domäne beziehungsweise Ihres Rechners enthalten ist. Der Einfachheit halber wird im folgenden Text auf die Benennung der Domäne verzichtet und ein Benutzer lediglich mit dem Namen der Windows-Gruppe beschrieben.

Per Doppelklick auf den Benutzer **Personal** öffnen Sie dessen Eigenschaften-Dialog. Auf der Seite **Mitgliedschaft** entfernen Sie die Häkchen zu den Datenbankrollen **db_datareader**, **db_datawriter** und **edb_execute**. Mit

einem Klick auf **OK** entziehen Sie dem Benutzer jegliche Zugriffsrechte.

Dem Benutzer **Vertrieb** nehmen Sie die Rechte auf ähnliche Weise. Öffnen Sie dessen **Eigenschaften**-Dialog und beenden Sie wie eben die Mitgliedschaft des Benutzers zu den Datenbankrollen **db_datareader**, **db_datawriter** und **edb_execute**. Jetzt ist noch die gesonderte Rechtevergabe des Benutzers aufzulösen. Dazu gehen Sie zur Seite **Sicherungsfähige Elemente**. Markieren Sie als erstes den Eintrag zur Tabelle **Bewerber** und entfernen Sie im unteren Bereich in der Spalte **Verweigern** die Häkchen in den Zeilen **Aktualisieren**, **Auswählen**, **Einfügen** und **Löschen** (siehe Bild 1). Wiederholen Sie den Vorgang für die Tabellen **Mitarbeiter** und **Stellen**. Danach markieren Sie die gespeicherte Prozedur **pSelectGeburtstagsliste** und deaktivieren die Rechtevergabe **Verweigern** zum Recht **Ausführen**. Bestätigen Sie die Änderungen mit einem Klick auf **OK**. Der Benutzer **Vertrieb** besitzt nun ebenfalls keinerlei Zugriffsrechte mehr. Um die Benutzer **Einkauf** und **Buchhaltung** müssen Sie sich nicht kümmern. Diese Benutzer haben aktuell noch keine Rechte.

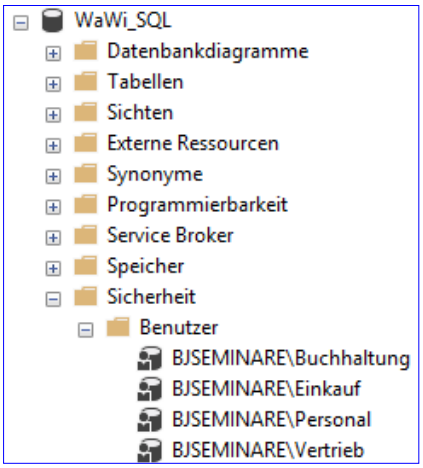

Bild 2: Die Benutzer in der Datenbank

Nun geht es an das Erstellen der neuen Datenbankrollen. Den Datenbankrollen mit den Datenbankobjekten und Zugriffsrechten, die die Anwender zum Erfüllen ihrer Aufgaben benötigen. Den Inhalt dieser Datenbankrollen gilt es zu ermitteln.

In der Beispielumgebung ist die Access-Applikation **WaWi v2** die einzige Applikation zur Datenbank. Somit findet hier die Bestandsaufnahme für das Berechtigungskonzept statt. Nach einem Start der Beispielapplikation zeigt Ihnen das Start-Formular die Schaltflächen mit den Funktionen für die Abteilungen **Vertrieb**, **Einkauf**

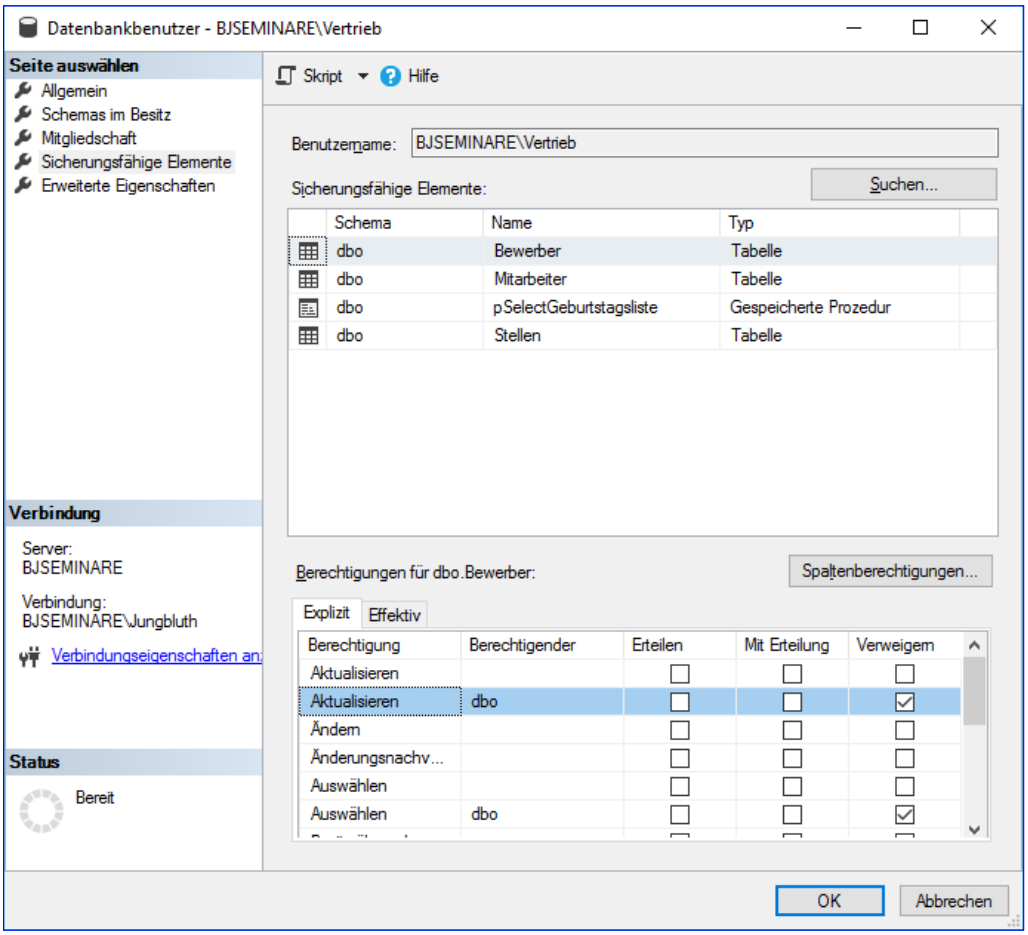

Bild 1: Entfernen des Rechts **Verweigern**

und **Personal** (siehe Bild 3). Auf den ersten Blick lässt dies auf drei Datenbankrollen schließen. Aber ist es so einfach? Reichen drei Datenbankrollen für eine gute Funktionstrennung und somit zur Vermeidung eines Datenmissbrauchs? Sie kennen die Antwort bereits. Solange Mitarbeiter die Daten der Lieferanten und die der Bestellungen mitsamt den Wareneingängen verwalten können, ist die Möglichkeit eines Datenmissbrauchs gegeben. Dieser lässt sich durch eine zusätzliche Aufteilung der Aufgaben in eine weitere Datenbankrolle vermeiden.

Unabhängig von den Missbrauchsmöglichkeiten steht noch eine Erweiterung der Zugriffsrechte an. Die Mitarbeiter haben sich bereit erklärt, dass jeder im Unternehmen die Geburtstagsliste einsehen darf. Bisher ist dies nur den Mitarbeitern im Personal erlaubt.

Dann werden es wohl mehr als drei Datenbankrollen. In den nächsten Schritten legen Sie die einzelnen Datenbankrollen an, bestimmen pro Datenbankrolle die Datenbankobjekte und definieren die erforderlichen Zugriffsrechte. Oder bei dem Beispiel von eben zu bleiben: Sie öffnen den Werkzeugkasten einer Abteilung und legen das benötigte Werkzeug hinein. Doch vorher brauchen Sie den Werkzeugkasten.

Datenbankrolle Personalwesen

Ran ans Werk. Beginnen Sie mit der Datenbankrolle für die Personalabteilung. Dazu navigieren Sie im SQL Server Management Studio in der Datenbank **WaWi_SQL** über **Sicherheit\Rollen** zum Eintrag **Datenbankrollen**. Klicken Sie mit der rechten Maustaste auf den Eintrag und wählen Sie den Befehl **Neue Datenbankrolle**.

Im folgenden Dialog geben Sie als erstes die Bezeichnung **Personalwesen** in das Feld **Rollenname** ein. Achten Sie bei der Namensvergabe darauf, dass nicht bereits ein Benutzer mit dem Namen existiert. Datenbankrollen und Benutzer sind im SQL Server beides Prinzipale. Sie können einen Namen entweder für eine Datenbankrolle oder für einen Benutzer vergeben. Eine doppelte Vergabe wird mit einer Fehlermeldung quittiert. So gesehen könnten Sie ebenso den Rollennamen **Personal** verwenden. Dieser Name wäre noch frei, da die Bezeichnungen der Benutzer den Domänennamen enthalten. Allerdings wird der Einfachheit halber in diesem Text der Benutzer ohne Domänenname beschrieben. Um nun Missverständnisse mit dieser verkürzten Schreibweise und einer gleichlautenden Datenbankrolle zu vermeiden, erhält die Datenbankrolle die Bezeichnung **Personalwesen**.

Bei dem Besitzer der Datenbankrolle folgen Sie dem Grundsatz KISS. Halten Sie es einfach und geben Sie das Kürzel **dbo** ein. **dbo** steht für **database owner** und somit ist der Besitzer der Datenbankrolle identisch mit dem der Datenbank. Es ist ebenso möglich, als Besitzer einen anderen Benutzer einzutragen. Dies kommt jedoch selbst in komplexeren Berechtigungskonzepten eher selten vor.

Anschließend klicken Sie auf die Schaltfläche **Hinzufügen** und wählen im folgenden Dialog über **Durchsuchen** den Benutzer **Personal** aus (siehe Bild 4). Ein Klick auf **OK** übernimmt Ihre Auswahl in die Vorauswahl, ein weiterer Klick auf **OK** in den Dialog zur Datenbankrolle. Jetzt gerade eben haben Sie die Anwender definiert, die mit der Datenbankrolle oder um beim Vergleich zu bleiben mit dem Werkzeugkasten **Personalwesen** arbeiten dürfen.

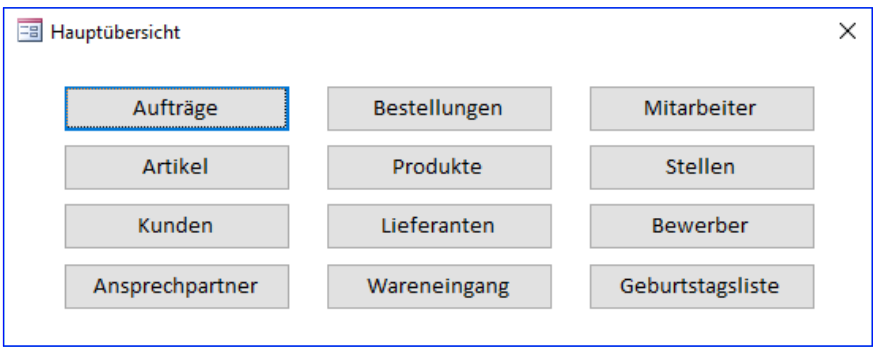

Bild 3: Das Start-Formular der Beispielapplikation

Fehlt noch das Werkzeug selbst, sprich die Datenbankobjekte mitsamt den Zugriffsrechten. Diese legen Sie in der Seite **Sicherungsfähige Elemente** fest. Doch welche Objekte benötigen die Mitarbeiter der Personalabteilung? Das zeigt die Bestandsaufnahme in der Beispielapplikation. Das Start-Formular enthält auf der rechten Seite die Schaltflächen für die Personalabteilung. Hierüber lassen sich die Daten der Mitarbeiter und Bewerber sowie neuer Stellenausschreibungen verwalten. Die Datenpflege erfolgt über Access-Formulare. Diese wiederum sind an Datenquellen gebunden. Genau diese Datenquellen bieten die Grundlage zur Rechtevergabe.

Beginnen Sie die Analyse mit den Mitarbeiterdaten. Dazu öffnen Sie mit der Schaltfläche **Mitarbeiter** das Formular zur Mitarbeiterverwaltung. Im Formular wechseln Sie in die Entwurfsansicht und aktivieren das Eigenschaftenblatt. Die Datenquelle des Formulars finden Sie in der Eigenschaft **Datensatzquelle** der Registerkarte **Daten**. Dort sehen Sie den Eintrag **Mitarbeiter** (siehe Bild 5).Das kann nun eine lokale Tabelle, eine eingebundene Tabelle, eine Access-Abfrage oder eine Pass Through-Abfrage sein. Was sich genau dahinter verbirgt, verrät Ihnen der Navigationsbereich. Eine Suche im Navigationsbereich zeigt Ihnen den Eintrag **Mitarbeiter** in der Rubrik **Tabellen**.

Anhand des Symbols ist es als eingebundene Tabelle zu erkennen (siehe Bild 6). Oder ist es eine eingebundene Sicht? Eine Antwort auf diese Frage werden Sie in Access nicht finden. Access behandelt eine eingebundene Sicht wie eine Tabelle. Diese Frage lässt sich nur im SQL Server Management Studio beantworten. Dort ist das Datenbankobjekt **Mitarbeiter** entweder bei den Tabellen oder bei den Sichten zu finden. Zum Glück müssen Sie das Objekt nicht mühsam in den Ordnern des Objekts-Explorers suchen. Der Dialog der Datenbankrolle bietet Ihnen eine Suche über mehrere Objekttypen.

Gehen Sie also zurück zum Dialog **Datenbankrolle – Neu** und wechseln Sie zur Registerkarte **Sicherungsfähige Elemente**. Hier klicken Sie auf **Suchen** und übernehmen in dem folgenden Dialog die vorgewählte Option **Bestimmte Objekte**. Es öffnet sich ein weiterer Dialog. Ein Klick auf die Schaltfläche **Objekttypen** bringt Sie zur

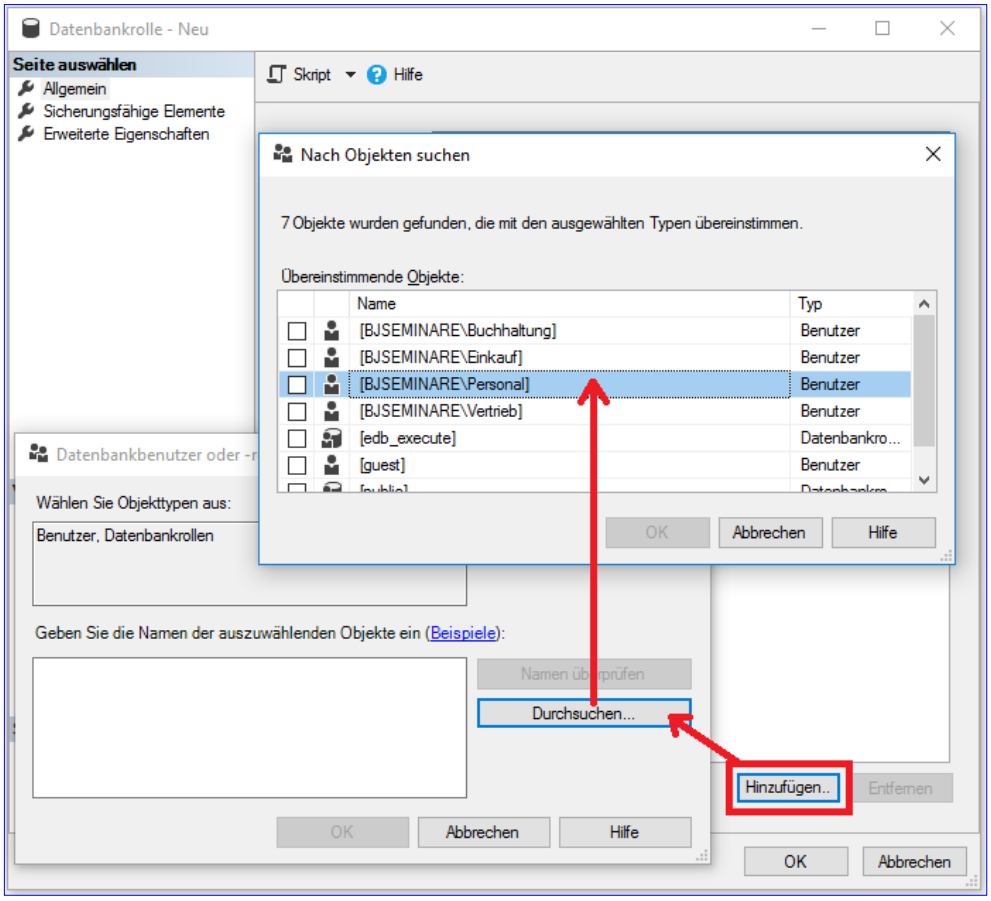

Bild 4: Auswählen des Benutzers **Personal**

nächsten Auswahlmöglichkeit. Dort aktivieren Sie die beiden Einträge **Tabellen und Sichten** und bestätigen diese mit **OK**. Zurück im Dialog **Objekte auswählen** geben Sie in das Eingabefeld den Wert **Mitarbeiter** ein und klicken auf **Namen überprüfen**. Sie erhalten einen weiteren Auswahldialog, der die Tabelle **Mitarbeiter** enthält (siehe Bild 7). Aktivieren Sie den Eintrag und übernehmen Sie die Auswahl mit **OK**. Den Dialog **Objekte auswählen** bestätigen Sie ebenfalls mit **OK**.

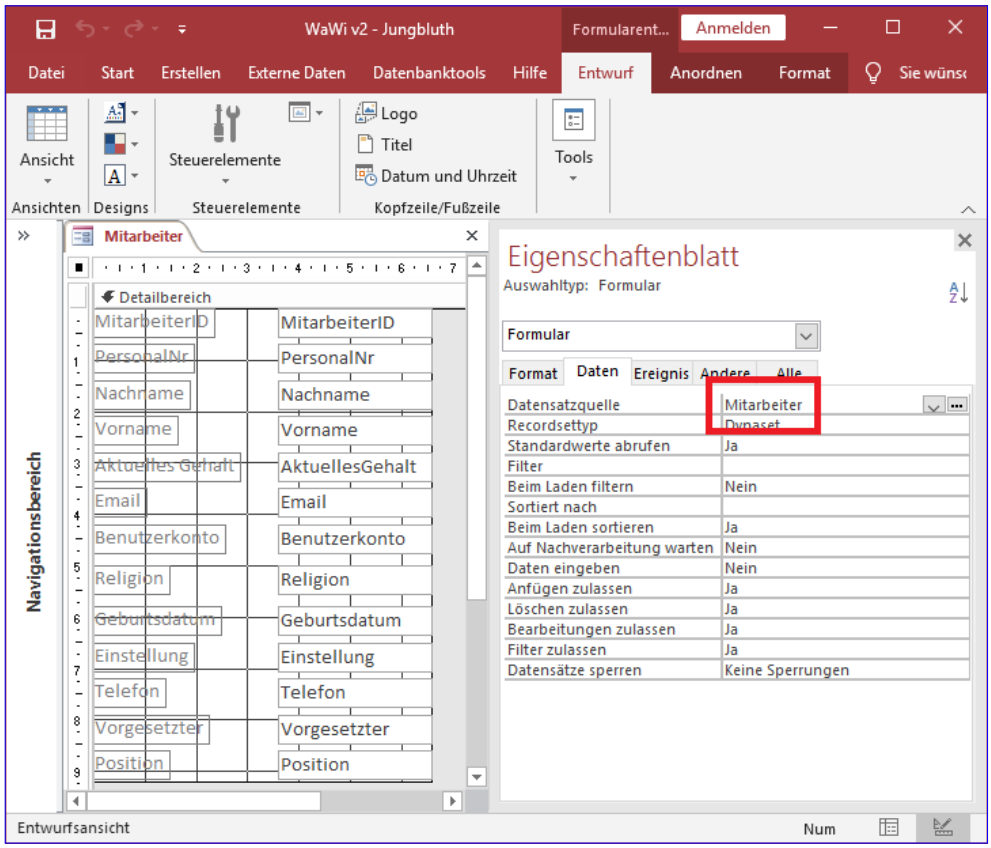

Bild 5: Die Datenquelle **Mitarbeiter**

Zurück im Dialog zur Datenbankrolle sehen Sie nun die Tabelle **Mitarbeiter** in der oberen Auflistung. Die Zugriffsrechte zur Tabelle definieren Sie im unteren Bereich. Die Beispielapplikation erlaubt das Lesen, Anlegen, Ändern und Löschen der Mitarbeiterdaten. Zur Vergabe dieser Rechte aktivieren Sie als erstes in der Zeile **Auswählen** die Spalte **Erteilen**. Hierüber wird das Recht **SELECT** zum Lesen der Daten dieser Tabelle vergeben. Auf die gleiche Weise aktivieren Sie mit einem Klick in **Erteilen** bei den Zeilen **Einfügen** und **Löschen** die Rechte **INSERT** und **DELETE**. Seien Sie vorsichtig bei der Vergabe des Rechts zum Ändern der

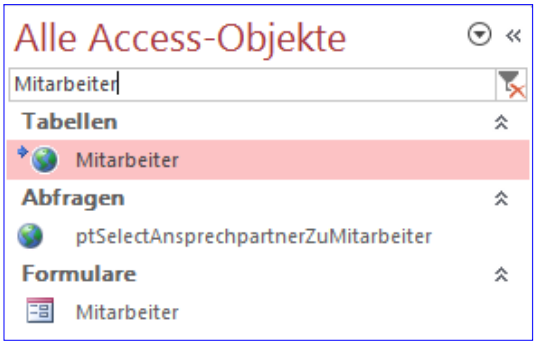

Bild 6: Die eingebundene Tabelle **Mitarbeiter**

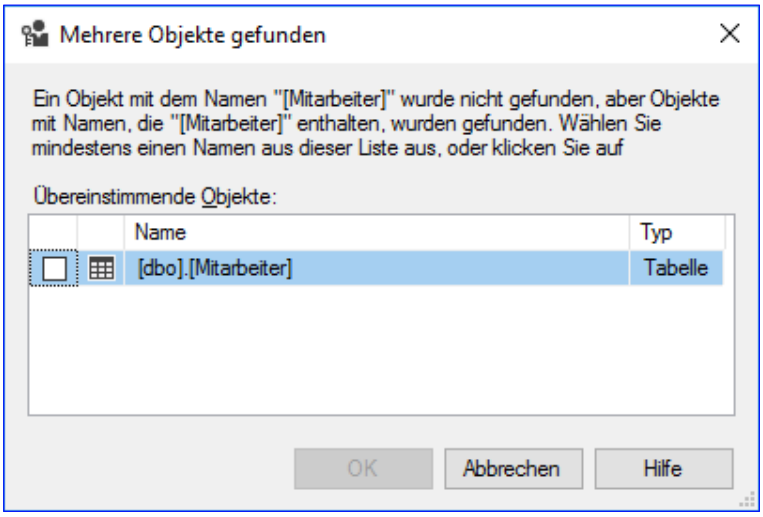

Bild 7: Auswählen der Tabelle **Mitarbeiter**

Daten. Dieses Recht wird gerne in der Zeile **Ändern** vergeben. Allerdings steht **Ändern** in diesem Fall für den Befehl **ALTER**, also das Recht, die Struktur der Tabelle zu ändern beziehungsweise diese zu löschen. Zur Vergabe des Rechts **UPDATE** müssen Sie die Option **Erteilen** in der Zeile **Aktualisieren** aktivieren.

So weit so gut. Das waren allerdings nur die Zugriffsrechte zu einer Schaltfläche der Beispielapplikation. Allein für die Personalabteilung gibt es dort noch drei weitere. Was im Vergleich zu vielen Access-Applikationen eher lachhaft ist. Zu jeder dieser Schaltflächen ist nun die Datenquelle zu ermitteln und im Dialog zur Datenbankrolle die entsprechenden Zugriffsrechte zu vergeben. Bei dem Klickaufwand wird das mit dem Dialog eine mühsame und zeitaufwendige Angelegenheit.

Sie haben die letzten Beiträge dieser Reihe gelesen? Dann wissen Sie was jetzt kommt. Klicken Sie im Dialog zur Datenbankrolle auf die Schaltfläche **Skript** (siehe Bild 8). Dieser Klick öffnet im SQL Server Management Studio eine neue Registerkarte und füllt diese mit T-SQL-Anweisungen.

Mit diesen Anweisungen legen Sie später die Datenbankrolle mitsamt den Mitgliedern und den eben vergebenen Rechten zur Tabelle **Mitarbeiter** an. Dieses Skript ist die Basis für die gesamte weitere Rechtevergabe.

Daher speichern Sie zunächst das Skript unter **Rechtevergabe WaWi_SQL** und beenden dann den Dialog zur Datenbankrolle mit **Abbrechen**. Jetzt räumen Sie das Skript ein wenig auf. Fassen Sie die einzelnen Zeilen zur Rechtevergabe der Tabelle **Mitarbeiter** in einer Zeile zusammen und verschieben Sie die Aufnahme des Benutzers in die Datenbankrolle an das Ende des Skripts:

```
USE WaWi_SQL;
GO
CREATE ROLE Personalwesen AUTHORIZATION dbo;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.Mitarbeiter TO
Personalwesen;
ALTER ROLE Personalwesen ADD MEMBER [BJSEMINARE\Personal];
GO
```

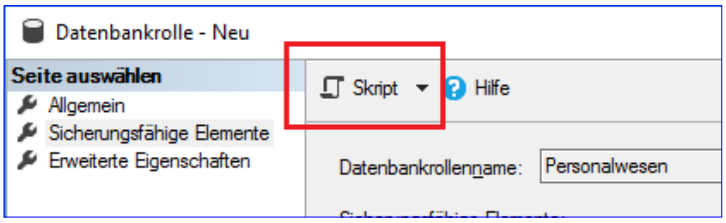

Bild 8: Erstellen des Rechte-Skripts

Der neue Aufbau des Skripts folgt im Gegensatz zum Dialog eher der anfangs beschriebenen Vorgehensweise. Die T-SQL-Anweisung **CREATE ROLE** erstellt die Datenbankrolle, also den Werkzeugkasten. Das Werkzeug selbst wird mit der Anweisung **GRANT** in den Werkzeugkasten gelegt, sprich die Tabelle **Mitarbeiter** mitsamt den Zugriffsrechen **SELECT**, **INSERT**, **UPDATE** und **DELETE** der Datenbankrolle zugeordnet. Zum Abschluss erlaubt die Anweisung **ALTER ROLE** einer oder mehreren Personen die Erlaubnis mit dem Werkzeugkasten zu arbeiten, in diesem Fall durch die Zuordnung des Benutzers **Personal** zur Datenbankrolle **Personalwesen**.

Das Skript werden Sie nun nach und nach mit den T-SQL-Anweisungen für die weiteren Datenbankrollen und deren Zugriffsrechte erweitern. Am Ende erhalten Sie die gesamte Rechtevergabe der Datenbank in einer übersichtlichen und lesbaren Form. Ergänzt mit aussagekräftigen Kommentaren ergibt sich daraus gleichzeitig die Dokumentation Ihres Berechtigungskonzepts (siehe Bild 9).

Nicht nur das. Jedes Mal, wenn Sie das Skript starten, werden die kompletten Zugriffsrechte der Datenbank erneut vergeben. So stellen Sie mit jeder Skriptausführung sicher, dass die Vorgaben Ihres Berechtigungskonzepts erfüllt sind.

Allerdings liefert die T-SQL-Anweisung **CREATE ROLE** derzeit noch einen Fehler, sollte die angegebene Datenbank-

rolle bereits existieren. Aus diesem Grund ergänzen Sie die T-SQL-Anweisung **CREATE ROLE** mit der folgenden **IF**-Anweisung.

```
IF (SELECT DATABASE_PRINCIPAL_ID('Personalwesen')) IS NULL
BEGIN
        CREATE ROLE Personalwesen AUTHORIZATION dbo;
END
```

Jetzt wird die Datenbankrolle nur angelegt, wenn die Funktion **DATABASE_PRINCIPAL_ID** keinen Wert zurückgibt. Wobei diese Prüfung etwas unscharf ist, liefert die Funktion doch die interne **ID** eines Prinzipals. Wie eben erwähnt, gehören hierzu neben den Datenbankrollen auch die Benutzer. Sollte es also einen Benutzer namens **Personalwesen** geben, ist das Ergebnis ungleich **NULL** und die Datenbankrolle wird nicht erstellt.

```
-------------------------------------------------------------
-- Rechtevergabe Datenbank WaWi_SQL
-- Erteilt am 20230710 vom Datenbankadministrator
-------------------------------------------------------------

USE WaWi_SQL;
GO

-------------------------------------------------------------
-- Rechtevergabe Personalwesen
-------------------------------------------------------------

-- Existenz der Datenbankrolle prüfen
IF (SELECT DATABASE_PRINCIPAL_ID('Personalwesen')) IS NULL
BEGIN
    -- Datenbankrolle Personalwesen anlegen
    CREATE ROLE Personalwesen AUTHORIZATION dbo;
END

-- Zugriffsrechte an Datenbankobjekten vergeben
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.Mitarbeiter TO Personalwesen;

-- Benutzer der Datenbankrolle zuordnen
ALTER ROLE Personalwesen ADD MEMBER [BJSEMINARE\Personal];
```

Bild 9: Die erste Version des Rechte-Skripts

Vielmehr erhält der Benutzer die Rechte zur Tabelle **Mitarbeiter** und die Anweisung **ALTER ROLE** erzeugt einen Fehler. Da in diesem Beispiel viel Wert auf die korrekte Benennung der Benutzer und Datenbankrollen gelegt wurde, soll an dieser Stelle die einfache Prüfung ausreichen. Schauen Sie mal in die Beispieldateien zu diesem Beitrag. Dort finden Sie in dem Skript **WaWi_SQL – Berechtigungskonzept** eine ausführlichere Version.

Die Anweisungen **GRANT** und **ALTER** ROLE benötigen keine vorherigen Prüfungen. Diese Aktionen lassen sich jederzeit wiederholen, selbst wenn die Rechte bereits erteilt und die Benutzer schon der Datenbankrolle zugeordnet sind.

Zurück zu den Zugriffsrechten für die Personalabteilung. Die weiteren benötigten Datenbankobjekte ermitteln Sie wieder in der Beispielapplikation. Hier gibt es noch die Schaltflächen **Bewerber** und **Stellen**. Den Weg zu den Datenquellen der zugehörigen Formulare kennen Sie bereits. Um es kurz zu machen: Dahinter verbergen sich die Tabellen **Bewerber** und **Stellen**. Die Mitarbeiter der Personalabteilung dürfen den Inhalt dieser beiden Tabellen lesen und ändern. Um diese Zugriffsrechte zu vergeben, kopieren Sie im Skript die folgende Zeile:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.Mitarbeiter TO Personalwesen;
```

Fügen Sie die Kopie unter der Originalzeile zweimal ein und ändern Sie in der ersten neuen Zeile den Tabellennamen in Bewerber und in der zweiten in Stellen. Aktuell besteht die Rechtevergabe innerhalb der Datenbankrolle aus diesen Zeilen:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.Mitarbeiter TO Personalwesen;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.Bewerber TO Personalwesen;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.Stellen TO Personalwesen;
```

SQL Server: Verschlüsselte Backups erstellen

Im Beitrag »SQL Server: Vollsicherung und Wiederherstellung« haben wir uns angesehen, wie wir eine Sicherung einer SQL Server-Datenbank durchführen können. Wenn wir uns eine solche Sicherungsdatei im Texteditor ansehen, stellen wir schnell fest, dass in einem Backup, wie wir es in diesem Beitrag vorgestellt haben, die Daten unverschlüsselt gespeichert sind. Liegen solche Daten im heimischen Server, der durch das Windows-Sicherheitssystem vor Zugriffen Fremder gesichert ist, mag das praktikabel sein. Sollen jedoch Kopien von diesen Daten angelegt werden, die nicht auf diese Weise gesichert sind, ist eine zusätzliche Verschlüsselung nötig. Das verschlüsselte Sichern einer SQL Server-Datenbank ist wesentlich aufwendiger als die einfache Sicherung. Deshalb schauen wir uns dies Schritt für Schritt im vorliegenden Beitrag an.

Im Beitrag **SQL Server: Vollsicherung und Wiederherstellung** (**www.access-im-unternehmen.de/1449**) haben wir uns bereits angesehen, wie wir unter dem SQL Server eine einfache Vollsicherung einer Datenbank durchführen können. Dort haben wir außerdem gezeigt, wie wir diese Sicherung zeitlich gesteuert automatisiert durchführen lassen können.

In einem weiteren Beitrag mit dem Titel **Zertifikate und Kennwörter mit Keepass speichern** (**www.access-im-unternehmen.de/1451**) haben wir das Tool Keepass vorgestellt, mit dem wir Daten wie Kennwörter oder Zertifikate speichern können. Dieses Tool werden wir in der folgenden Abhandlung nutzen, um die für die Wiederherstellung benötigten Daten sicher zu speichern. Denn was hilft eine verschlüsselte Kopie einer Datenbank, wenn die zum Entschlüsseln nötigen Informationen irgendwo auf der Festplatte herumliegen – oder noch besser auf einem Zettel an der Pinnwand?

Schlüssel und Zertifikat als Voraussetzung

Für das Verschlüsseln des Backups einer Datenbank benötigen wir einen Schlüssel und für diesen Schlüssel ein Zertifikat. Diese werden zum Verschlüsseln verwendet und auch zum Entschlüsseln, wenn eine Datenbank auf Basis einer verschlüsselten Datenbank wiederhergestellt werden soll. Dieses erstellen wir direkt im SQL Server Management Studio.

Den Schlüssel und das Zertifikat legen wir über T-SQL-Abfragen an, die wir im Kontext der **Master**-Datenbank des SQL Servers ausführen. Dazu öffnen wir im Objekt-Explorer des SQL Server Management Studios das Element

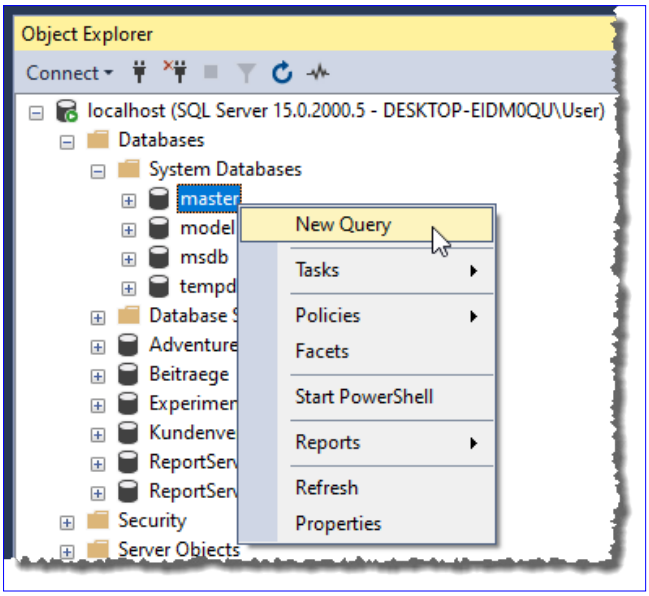

Bild 1: Neue Abfrage für die **Master**-Datenbank

Databases\|System Databases und klicken mit der rechten Maustaste auf den Eintrag **Master**. Im Kontextmenü führen wir den Befehl **New Query** aus (siehe Bild 1).

Master-Key für die Master-Datenbank

Als Erstes legen wir über eine T-SQL-Abfrage den sogenannten Master-Key für die Master-Datenbank an. Dazu nutzen wir den folgenden Befehl, mit dem wir den zu erstellenden Key angeben:

```
CREATE MASTER KEY
    ENCRYPTION BY PASSWORD = N'DiesIstDerSchluessel';
GO
```

Der **CREATE MASTER KEY**-Befehl erstellt einen Master-Key, der für die Verschlüsselung von Daten innerhalb der Datenbank verwendet wird. Der Schlüssel wird durch das angegebene Kennwort geschützt.

Das **N**-Präfix vor dem Kennwort gibt an, dass es sich um eine Unicode-Zeichenkette handelt.

Damit das Kennwort nicht leicht erraten werden kann, wollen wir es nicht wie im obigen Beispiel selbst vergeben, sondern das im oben angegebenen Beitrag **Zertifikate und Kennwörter mit Keepass speichern** verwendete Tool **Keepass** nutzen.

Nachdem wir dieses geöffnet haben, fügen wir als Erstes einen neuen Unterordner unter **Database** namens **SQLServer** hinzu. Hier sollen alle Elemente gespeichert werden, die mit dem SQL Server in Verbindung stehen. Dazu markieren wir den zuvor angelegten

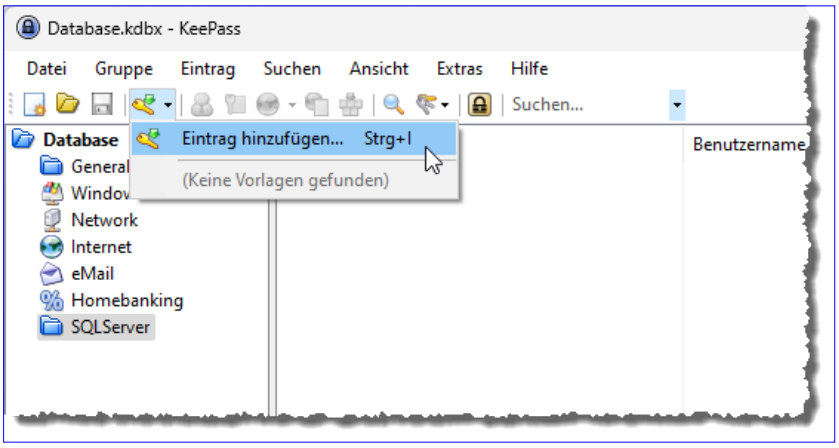

Bild 2: Einfügen eines neuen Eintrags in KeePass

Ordner **SQLServer** und wählen aus dem Menü den Befehl **Eintrag hinzufügen** aus (siehe Bild 2).

Danach öffnet sich der Dialog **Eintrag hinzufügen**. Hier geben wir als Titel beispielsweise **Masterkey für Daten-**

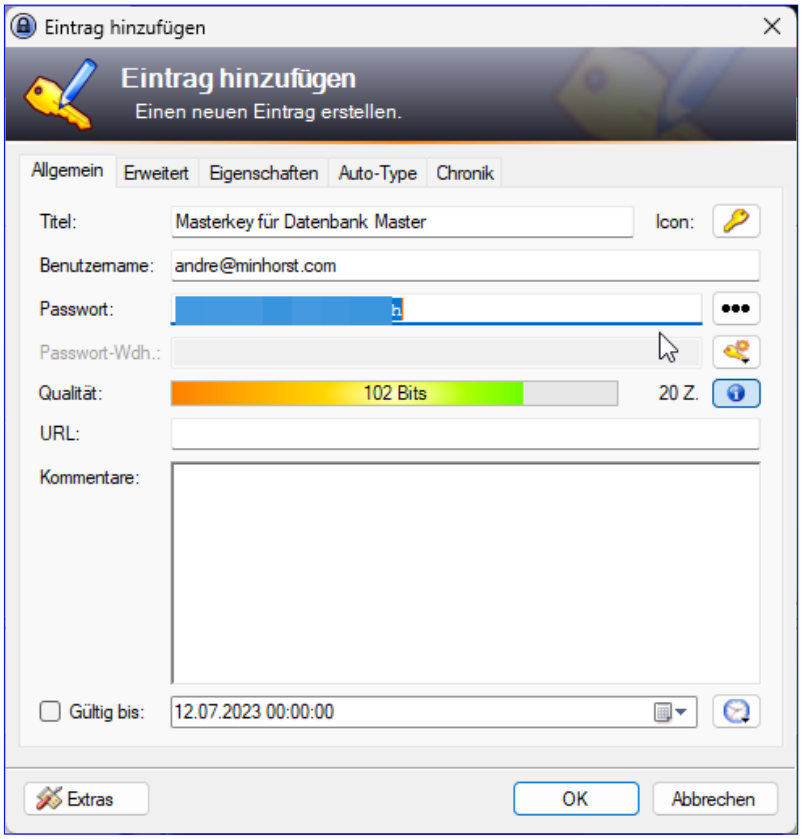

Bild 3: Erstellen eines neuen Kennworts

bank Master an. Außerdem lassen wir uns durch einen Klick auf die Schaltfläche mit den drei Punkten das bereits automatisch generierte Kennwort anzeigen (siehe Bild 3). Das ist notwendig, da wir es ja noch kopieren und in unser T-SQL-Skript einfügen wollen. Wenn Sie das Kennwort neu generieren lassen möchten, können Sie dazu die Schaltfläche **Ein Passwort generieren** nutzen, das rechts neben dem Feld **Passwort-Wdh.** angezeigt wird. Das Kennwort kopieren wir nun in die Zwischenablage.

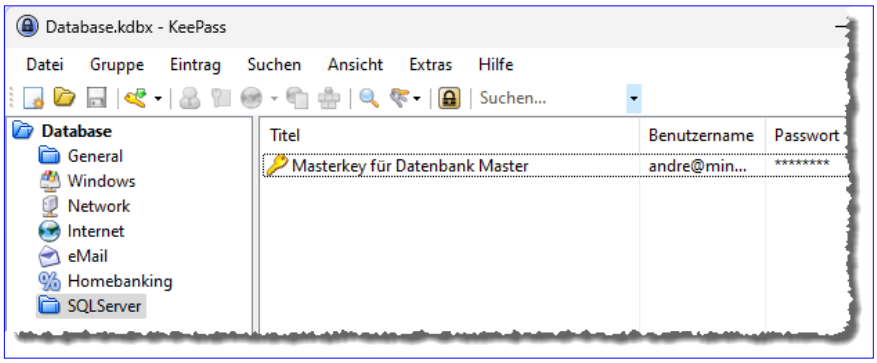

Bild 4: Der gespeicherte Schlüssel in **KeePass**

Nun folgt ein wichtiger Schritt – das Speichern des Kennworts in **KeePass**. Dazu schließen Sie den Dialog **Eintrag hinzufügen** und finden im Hauptfenster von **KeePass** den neuen Eintrag aus Bild 4 vor.

Master Key anlegen

Das Kennwort aus der Zwischenablage fügen wir nun wie folgt an Stelle des bisher verwendeten Kennworts **DiesIst-DerSchluessel** in der **CREATE MASTER KEY**-Anweisung ein:

```
CREATE MASTER KEY
    ENCRYPTION BY PASSWORD = N'PqM8BMLK5x7BBNejCxfh';
GO
```

Diesen Befehl können wir nun mit **F5** ausführen und erhalten die Meldung aus Bild 5 als Ergebnis.

Zertifikat erstellen

Nun wollen wir ein Zertifikat anlegen, das mit dem soeben erstellten Master-Key verschlüsselt wird. Dazu erstellen wir mit der folgenden Anweisung im gleichen Abfragefenster ein Zertifikat:

```
CREATE CERTIFICATE certBackup_Experimente
    WITH SUBJECT = N'Zertifikat zum Verschlüsseln der
Sicherung der Datenbank Experimente',
    EXPIRY_DATE = N'20401231';
```

Der **CREATE CERTIFICATE**-Befehl erstellt ein Zertifikat mit dem angegebenen Namen. Das Zertifikat wird verwendet, um Sicherungen der angegebenen Datenbank zu verschlüsseln. Das Zertifikat hat eine Beschreibung (**N'Zertifikat zum Verschlüsseln der Sicherung der Datenbank Experimente'**) und ein Ablaufdatum (**N'20401231'**).

Auch diesen Befehl führen wir nun aus. Damit der Befehl zum Erstellen des Schlüssels nicht ebenfalls ausgeführt wird, markieren wir nur den auszuführenden Befehl und betätigen anschließend die Taste **F5**, um diesen auszuführen.

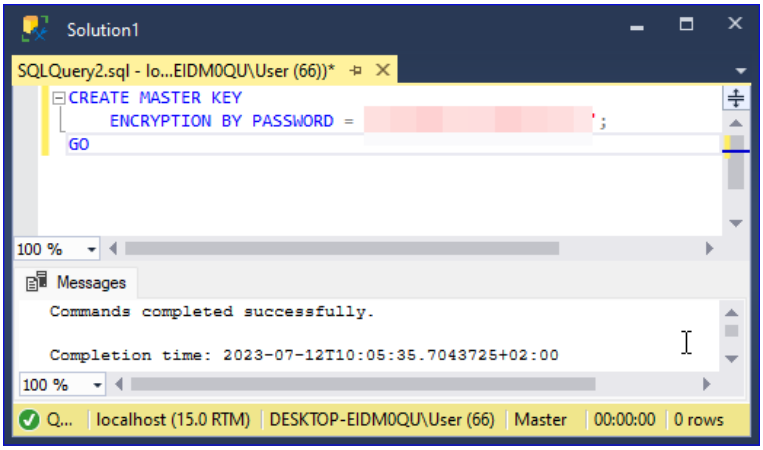

Bild 5: Anlegen eines Schlüssels für die **Master**-Datenbank

Damit haben wir das Zertifikat erstellt und es ist nun im Objekt-Explorer der **master**-Datenbank unter **SecuritylCertificates** zu finden (siehe Bild 6).

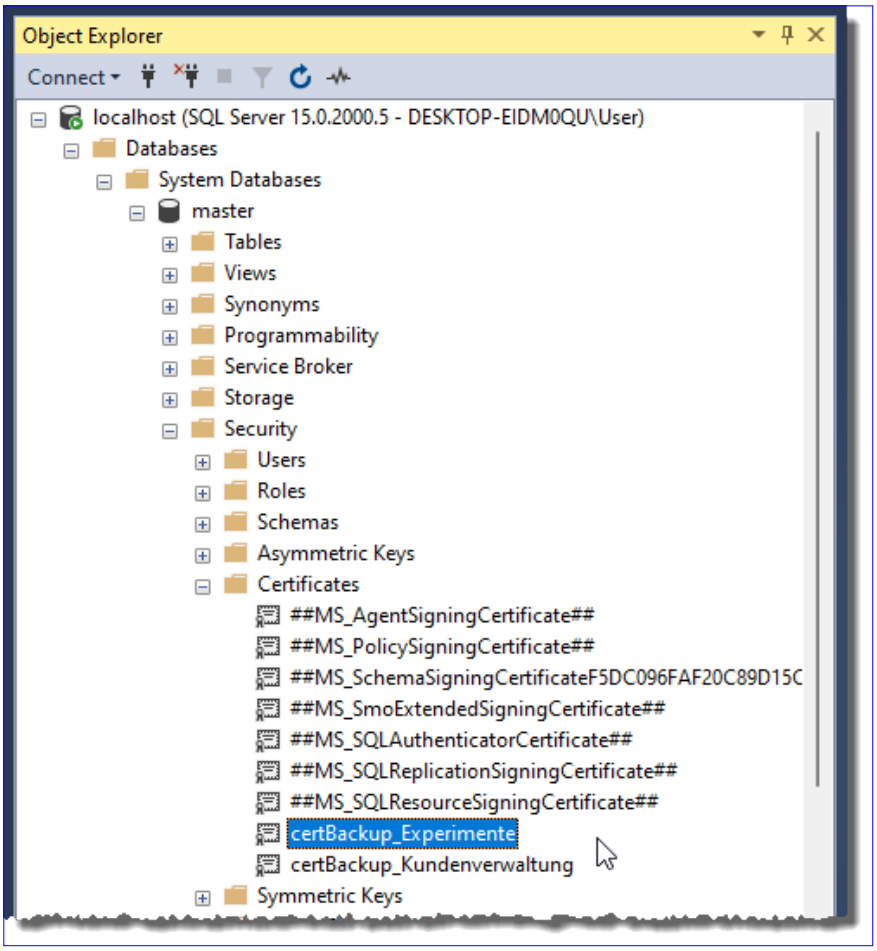

Bild 6: Das Zertifikat im Objekt-Explorer

Zertifikat und privaten Zertifikatsschlüssel als Datei sichern

Dieses Zertifikat können wir nun zur Verschlüsselung des Backups und zum Entschlüsseln beim Wiederherstellen des Backups verwenden. Der Haken ist jedoch: Wenn tatsächlich der komplette Rechner beschädigt wird und wir nicht mehr auf die SQL Server-Instanz zugreifen können, lässt sich auch die Sicherung der Datenbank nicht wiederherstellen.

Also müssen wir auch das Zertifikat sichern. Dazu nutzen wir einen weiteren T-SQL-Befehl, der wie in Listing 1 aussieht.

```
BACKUP CERTIFICATE certBackup_Experimente
    TO FILE = N'C:\...\certBackup_Experimente.cer'
    WITH PRIVATE KEY
        (ENCRYPTION BY PASSWORD = N'Zertifikatskennwort', FILE = N'C:\...\certBackup_Experimente.pvk');
GO
```

Listing 1: T-SQL-Skript zum Sichern des Zertifikats

Durch Ausführen dieses Skripts wird eine Sicherung des Zertifikats **certBackup_Experimente** erstellt, die aus der Zertifikatsdatei und der privaten Schlüsseldatei besteht.

Diese Dateien werden auf den angegebenen Speicherorten auf der Festplatte gespeichert. Die Sicherung kann später verwendet werden, um das Zertifikat und den privaten Schlüssel wiederherzustellen, falls dies erforderlich ist.

Der **BACKUP CERTIFICATE**-Befehl erstellt eine Sicherungskopie des Zertifikats mit dem Namen **certBackup_Experimente**. Das Zertifikat wird in zwei separate Dateien gesichert.

Die Zertifikatsdatei (**.cer**) wird angegeben durch den Parameter **TO FILE** mit dem Pfad **'C:\...\certBackup_Experimente.cer'**. Hier wird das Zertifikat im Zertifikatstransferformat gespeichert.

Die private Schlüsseldatei (**.pvk**) wird angegeben durch den Parameter **WITH PRIVATE KEY** mit dem Schlüssel **ENCRYPTION BY PASSWORD = N'Zertifikatskennwort'** und dem Dateinamen mit **FILE = N'C:\...\certBackup_Experimente.pvk'**.

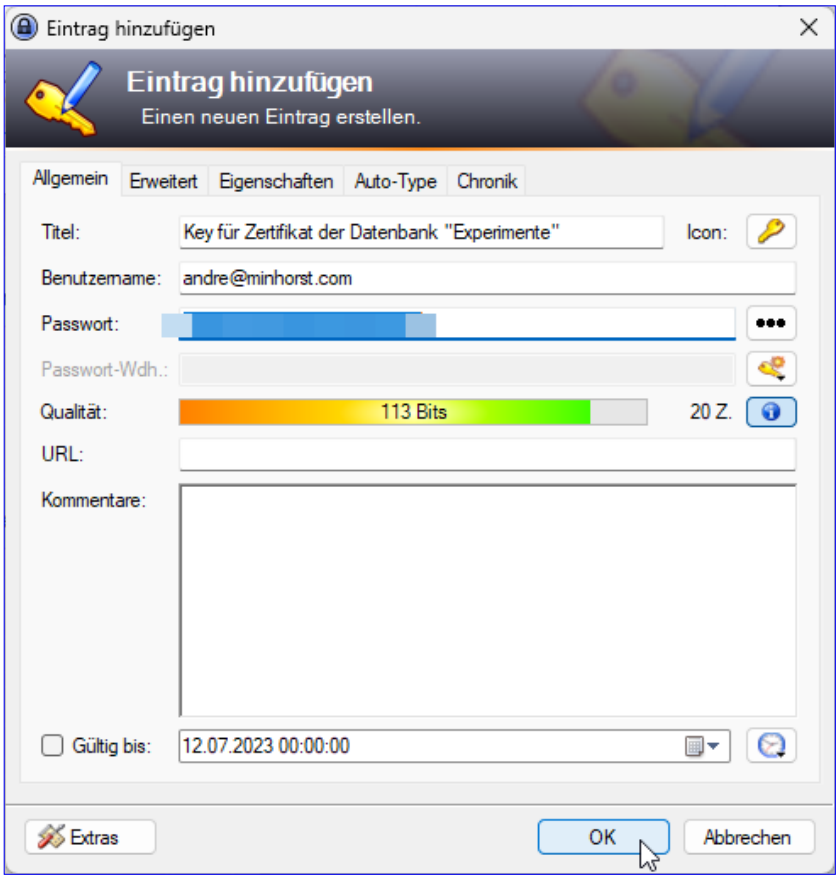

Bild 7: Anlegen eines privaten Schlüssels

Statt der drei Punkte in den beispielhaften Pfadangaben geben Sie ein Verzeichnis an, in dem die Dateien gesichert werden sollen. Aus diesem übertragen wir die Dateien später in **KeePass**. Es kann sein, dass dies einen Fehler liefert. Der Grund ist meist, dass der SQL Server keine Berechtigungen hat, in das angegebene Verzeichnis zu schreiben.

Um dies zu umgehen, ohne diese Berechtigungen entsprechend setzen zu müssen, verwenden wir ein Verzeichnis, in das der SQL Server auf jeden Fall schreiben kann – nämlich das Verzeichnis, in dem sich auch die Datenbankdateien befinden. Dieser Pfad lautet beispielsweise wie folgt:

```
C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA
```

Statt des hier verwendeten Kennwort-Platzhalters **Zertifikatskennwort** geben Sie ein neues Kennwort an, dass sie wieder mit Keepass erstellen. Dieses versehen wir wieder mit einem aussagekräftigen Titel, hier **Key für Zertifikat der Datenbank "Experimente"** (siehe Bild 7). Bevor wir dieses Kennwort durch Schließen des Dialogs speichern, kopieren wir es wieder in die Zwischenablage.

KeePass enthält nun zwei Einträge wie in Bild 8. Das Kennwort aus der Zwischenablage fügen wir für den Parameter **DECRYPTION BY PASSWORD** in den T-SQL-Befehl

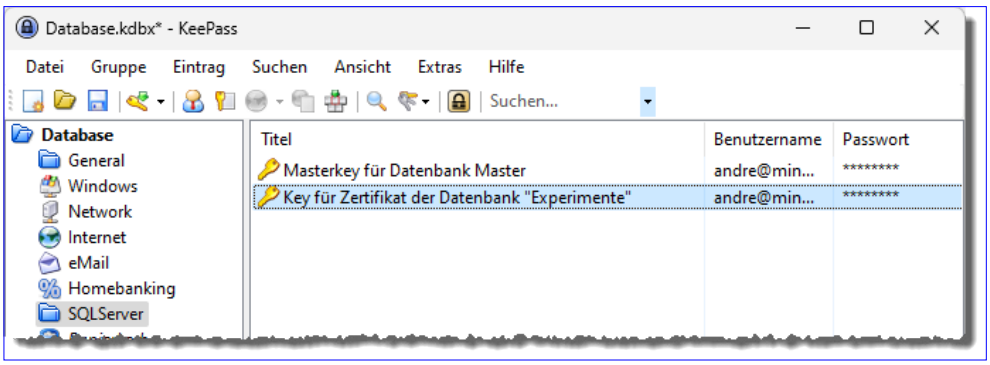

Bild 8: Der private Schlüssel für das Zertifikat in **KeePass**

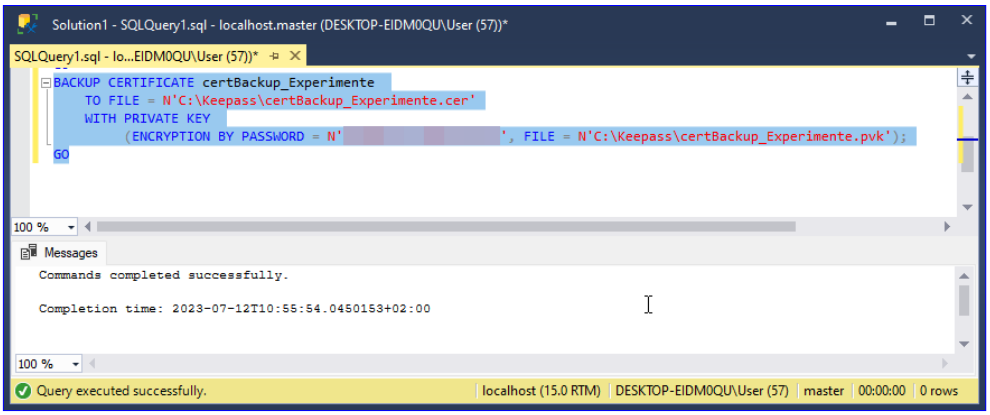

Bild 9: Sichern des Zertifikats auf der Festplatte

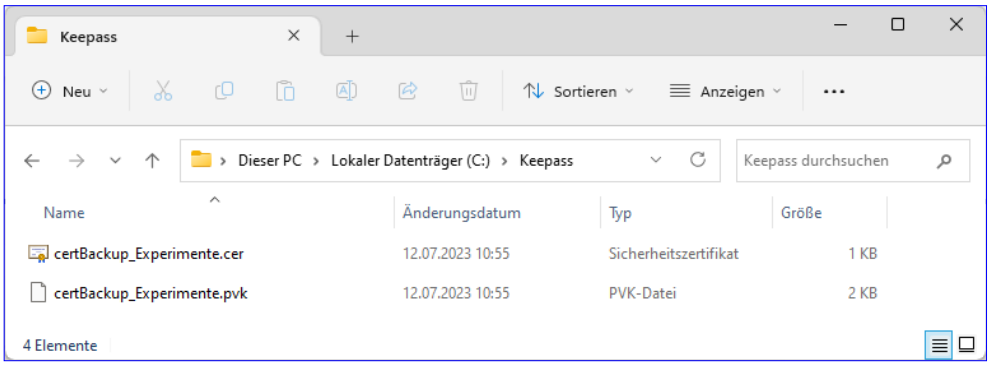

Bild 10: Das Zertifikat und die Schlüsseldatei auf der Festplatte

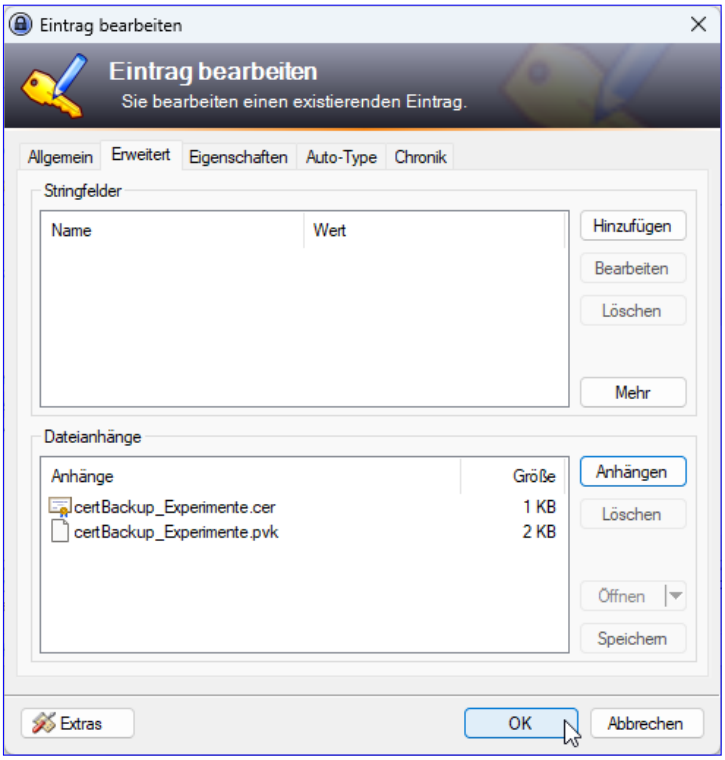

Bild 11: Sichern der Zertifikatsdatei und der Schlüsseldatei in **KeePass**

ein. Achten Sie beim Einfügen darauf, dass Sie nicht Leerzeichen oder andere nicht erwünschte Elemente zwischen den Hochkommata einfügen.

Den Befehl fügen wir wieder im Abfragefenster für die Datenbank **master** ein, markieren diesen und führen ihm mit **F5** aus, was wie in Bild 9 aussieht.

Nach dem Ausführen landen die beiden Dateien in dem angegebenen Verzeichnis – hier der Einfachheit halber **c:\Keepass** genannt (siehe Bild 10).

Nochmal zur Sicherheit: Diese Dateien dürfen nicht verloren gehen, denn sonst können Sie ein damit verschlüsseltes Backup einer Datenbank nicht wieder entschlüsseln.

Zertifikat und Schlüssel in KeePass sichern

Die Zertifikatsdatei und den Schlüssel haben wir zwar nun im Dateisystem gespeichert, allerdings hilft uns das auch nicht weiter, wenn der Rechner nicht mehr funktioniert oder anderweitig nicht mehr zugreifbar ist.

Also müssen wir diese auch noch separat sichern, und zwar so, dass diese Sicherung auch nach einem Verlust des Rechners noch wiederhergestellt werden kann.

In diesem Fall nutzen wir KeePass gleich noch ein zweites Mal: Wir öffnen wieder den Eintrag **Key für**

Schneller, weiter, höher mit LAA für Access 32-Bit

Es gibt einen Grund, warum man Access in der 64-Bit-Version gegenüber der 32-Bit-Version bevorzugen könnte, und das ist der virtuelle Arbeitsspeicher. Dieser beträgt bei der 64-Bit-Version satte vier Gigabyte, während die 32-Bit-Version nur zwei Gigabyte bietet. Dies wirkt sich vor allem dann aus, wenn viele Formulare oder Recordsets geöffnet sind – dann knallt es irgendwann einfach. Das lässt sich allerdings ändern, allerdings mit einem nicht ganz einfachen Eingriff: Es gibt in der MSAccess.exe genau ein Bit, das man ändern muss, damit auch die 32-Bit-Version über vier Gigabyte Arbeitsspeicher verfügt. Das ist erstens ohne Hilfsmittel nicht zu machen und zweitens wird diese Änderung wieder rückgängig gemacht, wenn ein Update eine neue Version der MSAccess.exe mitbringt. Also haben wir neben den Grundlagen zu diesem Problem, die wir in diesem Beitrag beschreiben, auch noch ein Tool mitgebracht, mit dem Sie die Performance von Access 32-Bit verbessern können.

Was anderen Office-Anwendungen wie Outlook oder Excel längst gewährt wurde, ist bei Microsoft Access immer noch nicht geschehen: Die 32-Bit-Version von Access wird standardmäßig mit nur zwei Gigabyte nutzbarem virtuellem Arbeitsspeicher ausgeliefert.

Und das, obwohl Microsoft eigentlich schon 2020 angekündigt hatte, dies zu tun. Nun denn: Mittlerweile gibt es Tipps und Tricks, wie man die Datei **MSAccess.exe** selbst so anpasst, dass diese vier Gigabyte Arbeitsspeicher nutzen kann.

Download des LAA-Tools

Das kostenlose Tool finden Sie unter dem folgenden Link:

`https://me.minhorst.com/amvLAA`

Das amvLAA-Tool

Das ist allerdings teilweise recht unhandlich, weshalb wir ein Tool entwickelt haben, das speziell diese Aufgabe übernimmt – und noch ein wenig mehr. Genau genommen handelt es sich bei diesem Tool um eine Kombination aus einem COM-Add-In und einer ausführbaren Datei.

Das COM-Add-In stellt die Funktionen rund um LAA in Access selbst zur Verfügung. Das schließt beispielsweise eine Prüfung beim Start von Access ein, ob die Erweiterung auf vier Gigabyte derzeit aktiviert ist oder nicht.

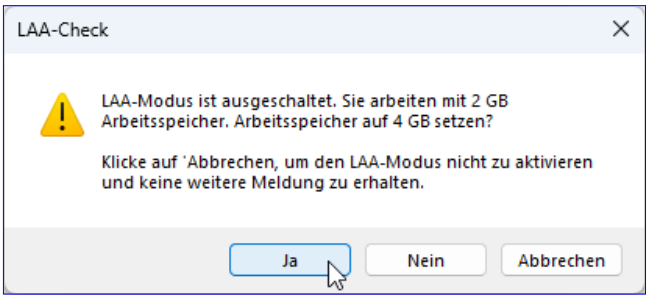

Bild 1: Meldung beim Start von Access, wenn LAA nicht aktiviert ist

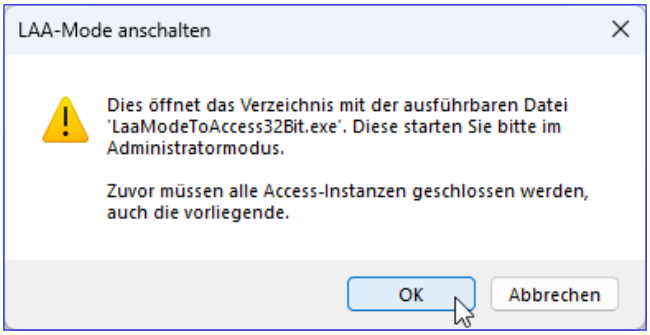

Bild 2: Anweisungen zum Setzen des LAA-Modus

Falls nicht, wird eine entsprechende Meldung angezeigt (siehe Bild 1).

Hier haben wir bereits alle Möglichkeiten, die wir benötigen:

- **Ja**: Startet den Vorgang, um den Arbeitsspeicher auf vier Gigabyte zu erhöhen.
- **Nein**: Schließt die Meldung und behält die Einstellungen bei.
- **Abbrechen**: Behält den Modus mit zwei Gigabyte Arbeitsspeicher bei und sorgt dafür, dass die Meldung nicht mehr angezeigt wird.

LAA-Modus mit vier Gigabyte Arbeitsspeicher aktivieren

Wenn man den LAA-Modus aktivieren und aktiviert halten möchte, klickt man hier einfach auf **Ja**. Dies führt dazu, dass eine weitere Meldung mit Anweisungen erscheint. Diese lauten, dass nun ein Explorer-Fenster geöffnet wird, das die zu startende **.exe**-Datei anzeigt (siehe Bild 2).

Im nun erscheinenden Windows Explorer finden wir die Datei **LaaModeToAccess32Bit.exe** vor. Diese starten wir nun per Doppelklick (siehe Bild 3).

Danach erscheint das Fenster aus Bild 4. Wenn Sie die Access-Instanz, von der aus der Windows Explorer mit dem Tool geöffnet wurde, noch nicht geschlossen haben oder eine andere Access-Instanz noch offen ist, zeigt das Tool dies direkt an.

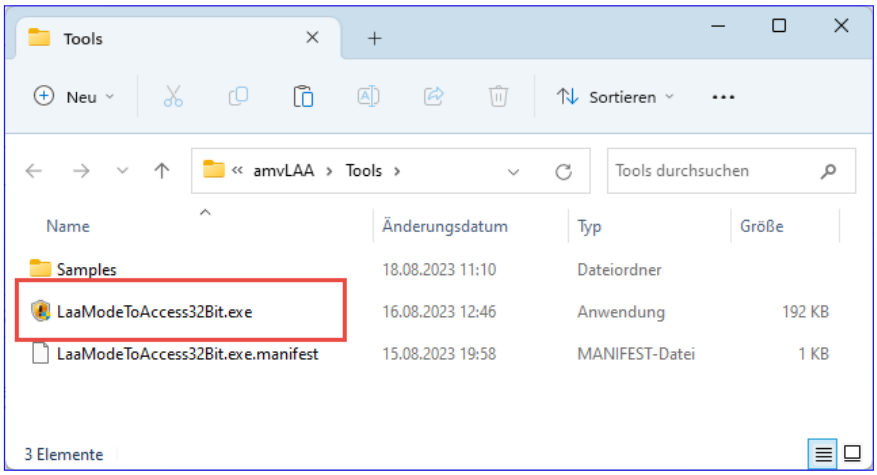

Bild 3: Die zu öffnende Datei **LaaModeToAccess32Bit.exe**

Bild 4: Hinweis, alle Access-Anwendungen zu schließen

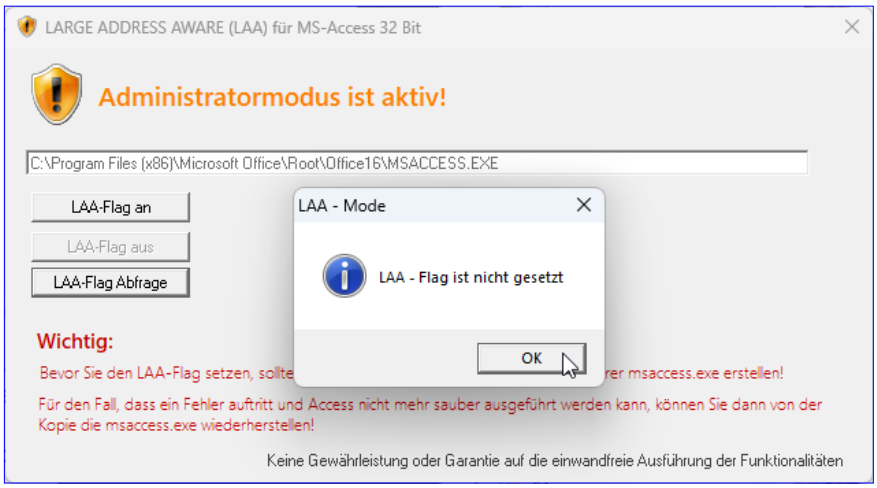

Bild 5: Anzeige des Status bezüglich des LAA-Flags

Um dies zu ändern, brauchen wir das Tool allerdings nicht neu zu starten – wir schließen einfach alle Access-Instanzen und das Tool erkennt es automatisch.

Ein Klick auf die Schaltfläche LAA-Flag-Abfrage liefert dann per Meldungsfenster den aktuellen Status (siehe Bild 5).

Nun können wir mit einem Klick auf die Schaltfläche LAA-Flag an den verfügbaren Speicher von zwei auf vier Gigabyte anheben.

amvLAA in Access

Beim nächsten Start erscheint keine Meldung mehr, die auf das nicht gesetzte LAA-Flag hinweist. Access startet wie gewohnt und zeigt keine besonderen Auffälligkeiten.

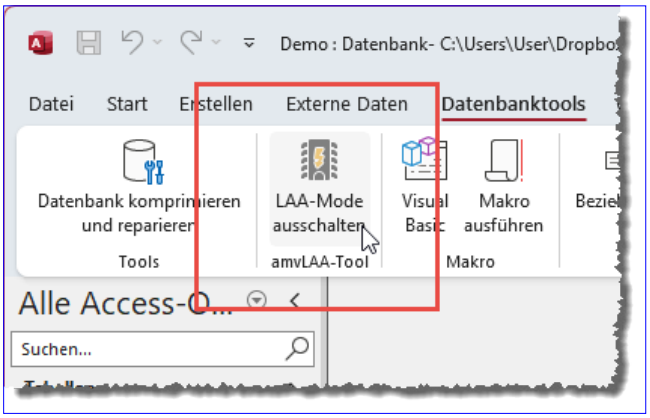

Bild 6: Ribbon-Befehl zum Aktivieren oder Deaktivieren des LAA-Modus

Mit Ausnahme derer, die wir über das COM-Add-In hinzugefügt haben. So finden wir beispielsweise im Ribbon im Tab **Datenbanktools** ein neues Element namens (siehe Bild 6).

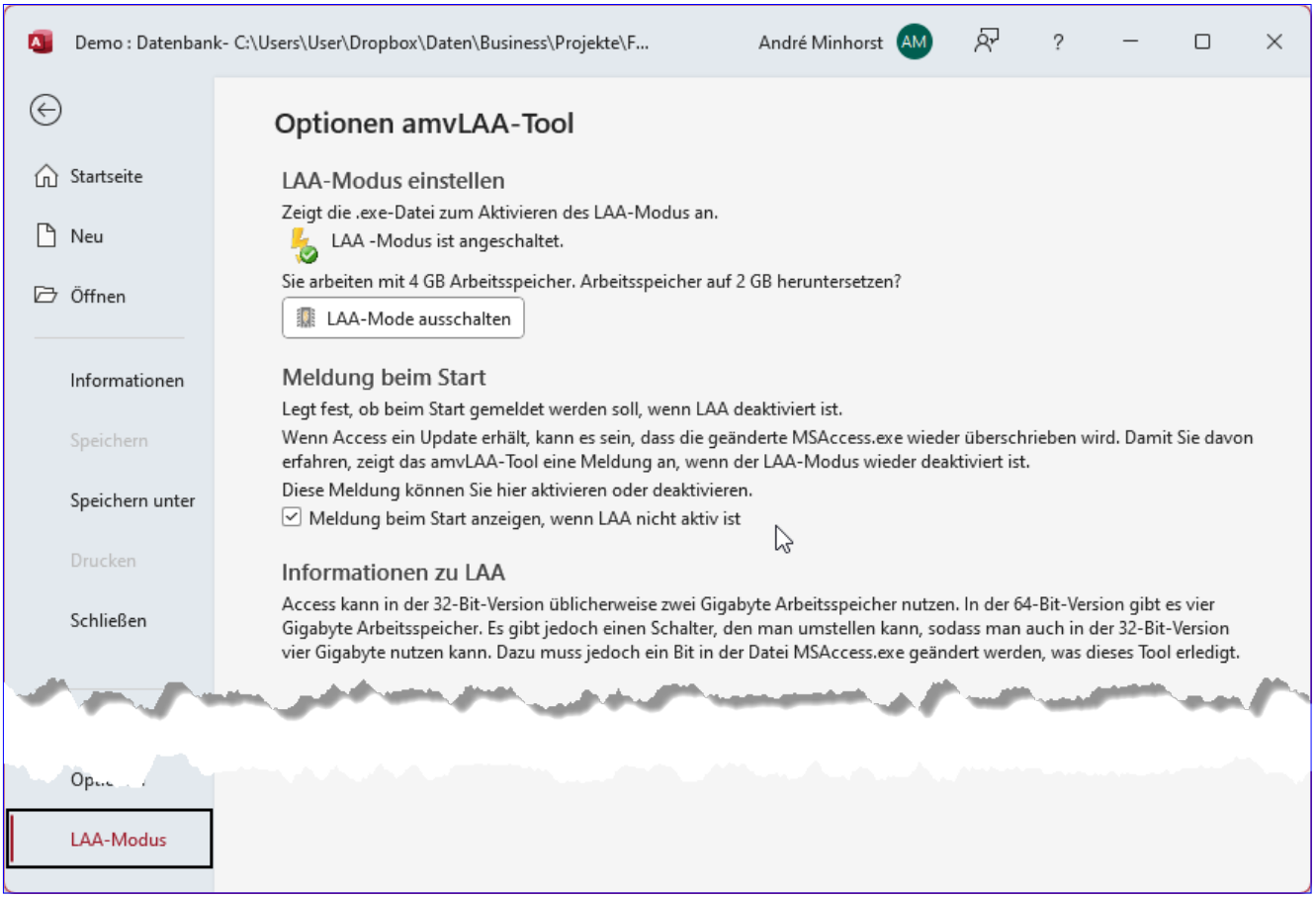

Bild 7: Einstellungen zum Tool im Datei-Menü

Außerdem finden wir im Datei-Menü einige Einträge mit Informationen und Einstellungen (siehe Bild 7). Hier finden wir im oberen Bereich den aktuellen Modus – in diesem Fall ist LAA aktiviert. Deshalb bietet die Schaltfläche darunter auch die Möglichkeit, diesen Modus wieder auszuschalten.

Schließlich können wir im unteren Bereich noch festlegen, ob wir beim Öffnen von Access eine Meldung angezeigt bekommen wollen, wenn der LAA-Modus nicht aktiviert ist. Ist diese Option aktiviert, erscheint die Meldung, wenn durch ein Update eine neue Version der Datei **MSAccess.exe** installiert wurde, bei der LAA nicht aktiviert ist.

Probleme ohne LAA-Modus

Wie schon erwähnt, stehen Access in der 32-Bit-Version nur zwei Gigabyte virtueller Arbeitsspeicher zur Verfügung – gegenüber vier Gigabyte bei der 64-Bit-Version.

Probleme treten nun beispielsweise auf, wenn ein Entwickler eine Access-Datenbank mit der 64-Bit-Version erstellt hat und dort größere Mengen an Formularen, Formularen mit Unterformularen oder Recordsets geöffnet wurden.

Dabei wird mit jeder geöffneten Instanz eines Formulars mehr von dem Speicher belegt, welcher der Access-Anwendung zugewiesen ist. Der verfügbare Speicher ist einigermaßen konstant, solange der Arbeitsspeicher des Rechners dies hergibt und nicht andere Anwendungen mit höherer Priorität diesen Speicher belegen.

Aber nicht nur geöffnete und somit in der Regel sichtbare Instanzen von Formularen fressen den verfügbaren virtuellen Speicher, sondern vor allem auch Zugriffe auf Daten.

Diese treten in der Form von Recordsets auf, die nicht nur in Zusammenhang mit angezeigten Daten geöffnet werden, sondern auch per VBA-Code. Hier sammeln sich schneller relevante Datenmengen an, als man denkt und so treten dann Fehler durch fehlende Ressourcen auf.

In einem weiteren Beitrag namens **Access-Speicher überwachen mit VMMap** (**www.access-im-unternehmen.de/1457**) schauen wir uns an, wie wir den aktuellen Speicher von Access untersuchen können und wie sich verschiedene Elemente auf den virtuellen Speicher auswirken.

Hier betrachten wir in einigen Beispielen auch, welche Elemente sich wie auf den Verbrauch des virtuellen Speichers auswirken und welche Fehler auftreten, wenn dieser ausgereizt wird.

Sollte so ein Fehler auftreten und man verwendet die 32-Bit-Version von Access, kann man sich durch Aktivieren des LAA-Modus eine Menge mehr virtuellen Speicher mehr verschaffen (auch hier wieder der Hinweis, dass der Arbeitsspeicher des Rechners dies hergeben muss).

LAA-Modus immer automatisch aktivieren

Wie zu Beginn erwähnt, kann es sein, dass Microsoft ein Update liefert, das auch die **MSAccess.exe** durch eine neue Version ersetzt. Wie oft das geschieht, variiert je nach der verwendeten Edition.

Aber gelegentlich geschieht es, und es wäre schade, wenn die Benutzer dadurch wieder in Fehler laufen, die wir durch das Setzen des LAA-Modus weitgehend ausgeschlossen haben.

Wir haben mit dem COM-Add-In bereits eine Technik vorgestellt, mit der wir beim Öffnen von Access darauf hingewiesen werden, wenn der LAA-Modus deaktiviert wurde. Allerdings kann es auch geschehen, dass das COM-Add-In aus irgendeinem Grund deinstalliert wurde oder wir die Meldung über die entsprechende Option deaktiviert haben. Oder wir wollen gar nicht, dass sich der Benutzer in diesem Fall mit dem erneuten Aktivieren des LAA-Modus beschäftigen muss.

Dann haben wir noch eine Alternative. Die Datei **LaaModeToAccess32Bit.exe** können wir nämlich auch in

Access-Speicher überwachen mit VMMap

Trotz eigentlich ausreichenden Arbeitsspeichers und sonstiger Systemressourcen kann es vorkommen, dass Access in die Knie geht. Das macht sich durch verschiedene Fehlermeldungen bemerkbar. Aber wo liegen eigentlich die Grenzen von Access in der 32-Bit- und der 64-Bit-Version und wie findet man heraus, wie viele Formulare oder Recordsets Access vertragen kann? Das schauen wir uns in diesem Beitrag einmal genauer an. Dabei nutzen wir ein Tool namens VMMap von Microsoft, mit dem wir uns den Speicherbedarf von Anwendungen wie Access genau ansehen können.

Werkzeug für diesen Beitrag: VMMap

VMMap ist ein hilfreiches Tool, um den virtuellen Adressraum und den Speicherverbrauch von Prozessen zu visualisieren und zu analysieren. Die verschiedenen Werte, die wir in VMMap sehen, repräsentieren verschiedene Arten von Speicherbereichen und -ressourcen im virtuellen Adressraum eines Prozesses.

- **Committed**: Der »Committed« (zugesagte) Speicher bezieht sich auf den Teil des virtuellen Adressraums eines Prozesses, der tatsächlich für die Verwendung im physischen Speicher oder in der Auslagerungsdatei reserviert ist. Es stellt den Speicher dar, den der Prozess für seine Operationen nutzen kann. Ein zugesagter Speicher muss nicht sofort im physischen RAM vorhan-

VMMap herunterladen

VMMap laden wir direkt beim Microsoft herunter. Eine zum Zeitpunkt der Erstellung dieses Beitrags gültige Downloadadresse lautet:

```
https://learn.microsoft.com/de-de/sysinternals/downloads/vmmap
```

Dies lädt das Programm **vmmap.exe** herunter.

VMMap starten

Starten wir die Datei **vmmap.exe**, erscheint der Dialog aus Bild 1. Hier wählen wir den Prozess aus, den wir hinsichtlich des Speicherbedarfs untersuchen wollen. In diesem Fall wählen wir den Eintrag **MSACCESS.EXE** aus.

Direkt im Anschluss wird es in dem nun geöffneten Fenster für diesen Prozess sehr bunt (siehe Bild 2). Hier sehen wir im oberen Bereich drei Balken, die folgende Bedeutung haben:

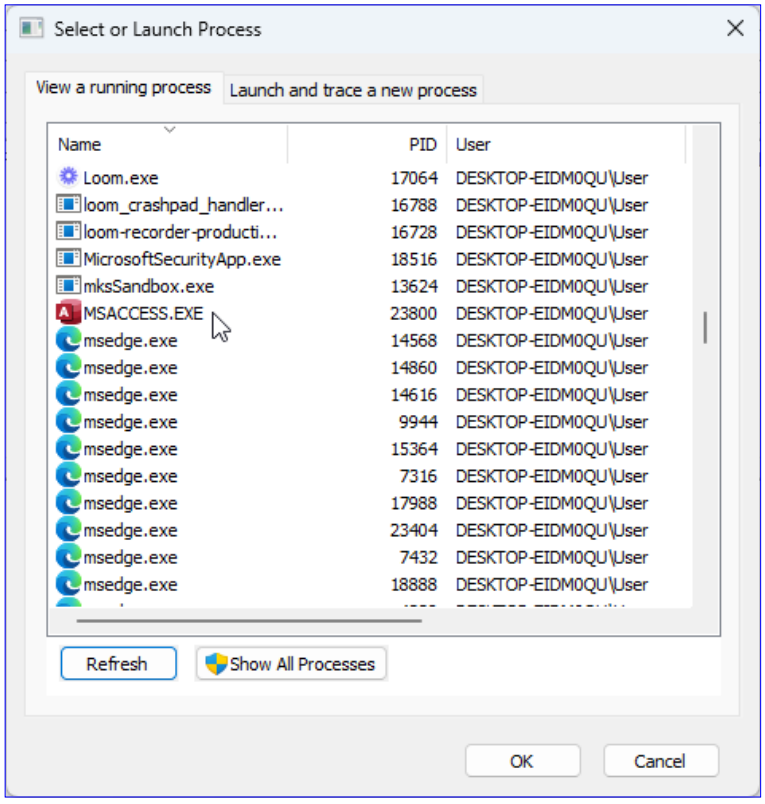

Bild 1: Übersicht der Prozesse

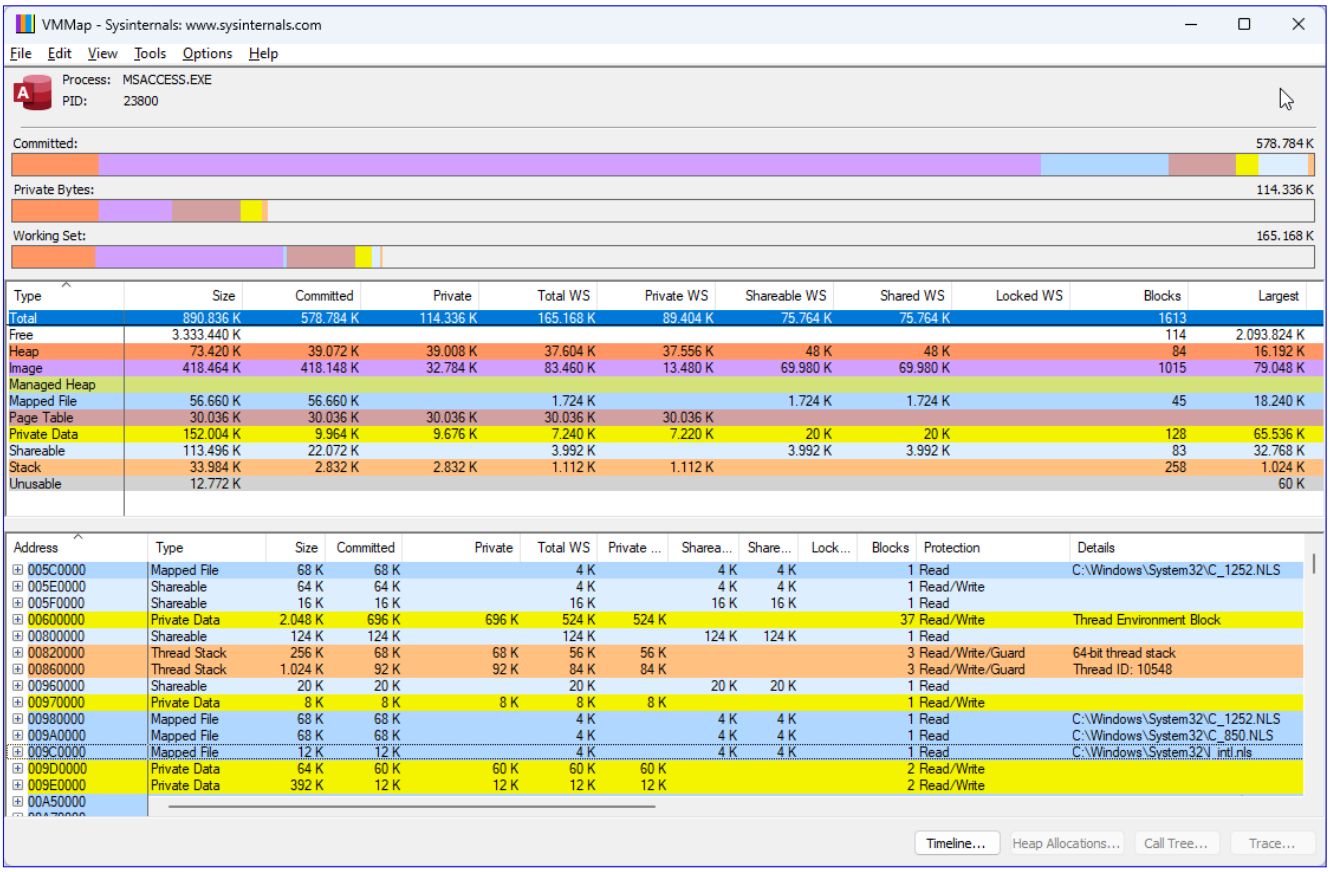

Bild 2: Daten eines Prozesses

den sein, sondern kann bei Bedarf in die Auslagerungsdatei verschoben werden.

- **Private Bytes**: »Private Bytes« sind der Teil des »Committed« Speichers, der nicht von anderen Prozessen oder Anwendungen geteilt wird. Dies ist der Speicher, der ausschließlich für den aktuellen Prozess reserviert ist. Private Bytes repräsentieren den Verbrauch von tatsächlichem physischem RAM und Auslagerungsdatei-Speicher durch den Prozess.

- **Working Set**: Das »Working Set« ist der tatsächliche physische Speicher (RAM), der derzeit von einem Prozess im Arbeitsspeicher gehalten wird. Es umfasst diejenigen Seiten im virtuellen Adressraum des Prozesses, die aktiv verwendet werden. Der **Working Set**-Wert ändert sich ständig, wenn der Prozess Daten liest und schreibt.

Die Balken stellen die darunter abgebildeten Informationen in einer anderen Ansicht dar.

Hier ist eine Erklärung für die darunter liegenden Zeilen:

- **Image**: Dieser Bereich enthält den ausführbaren Code und die Daten des Prozesses. Es umfasst das Programm selbst sowie die DLLs und anderen Dateien, die vom Prozess verwendet werden.

- **Heap**: Ein Heap ist ein Speicherbereich, in dem dynamisch zugewiesene Speicherblöcke für Datenstrukturen wie Arrays und Listen verwaltet werden. Dieser Bereich wächst und schrumpft je nach Bedarf.

- **Page Table**: Die Seitenverwaltungstabelle ist eine interne Struktur, die das Betriebssystem verwendet, um den physischen Speicher mit den virtuellen Adressen

zu verknüpfen. Dieser Bereich zeigt Informationen über die Seitenzuordnung.

- **Private Data**: Private Daten sind Bereiche im virtuellen Adressraum, die von diesem Prozess für seine eigenen Zwecke reserviert wurden. Sie sind für andere Prozesse nicht zugänglich.

- **Shareable**: »Shareable«-Bereiche sind solche, die zwischen verschiedenen Prozessen gemeinsam genutzt werden können, zum Beispiel durch Speicherzuweisungen über gemeinsam genutzte Bibliotheken oder Ressourcen.

- **Mapped File**: Dieser Bereich repräsentiert Dateien, die in den Speicher des Prozesses abgebildet (geladen) wurden. Diese Dateien können von anderen Prozessen oder dem Betriebssystem geändert werden.

- **Stack**: Der »Stack« ist ein Bereich des Speichers, der für die Verwaltung von Funktionen und Variablen in einem Programm verwendet wird. Er wächst normalerweise von oben nach unten und enthält Informationen über Aufrufer und Parameter von Funktionen.

- **Managed Heap**: Dieser Bereich ist spezifisch für verwalteten Code, normalerweise im Kontext von .NET-Programmierung. Er enthält Objekte, die vom .NET-Heap verwaltet werden.

- **Unusable**: »Unusable«-Bereiche sind Speicherbereiche, die nicht verwendet oder zugewiesen werden können. Dies kann beispielsweise aufgrund von Sicherheitsmaßnahmen oder Betriebssystemeinschränkungen der Fall sein.

- **Free**: Dies sind Bereiche, die im virtuellen Adressraum als nicht zugewiesener oder freier Speicherplatz markiert sind und derzeit nicht verwendet werden.

Der Bereich darunter schlüsselt die verschiedenen Bereiche sehr detailliert auf – diesen Bereich wollen wir uns nicht genauer ansehen. Interessant sind die Werte **Size** in der Zeile **Total** und **Free**.

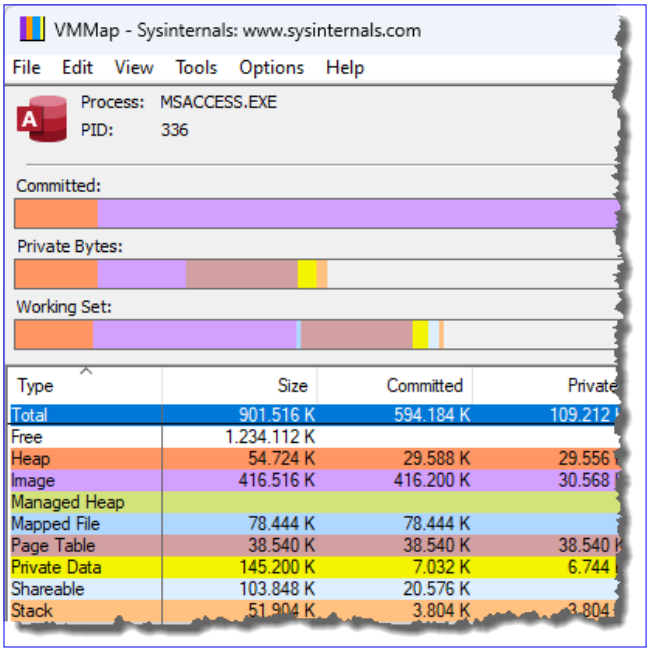

Bild 3: Speicherbedarf von Access ohne aktivierten LAA-Modus

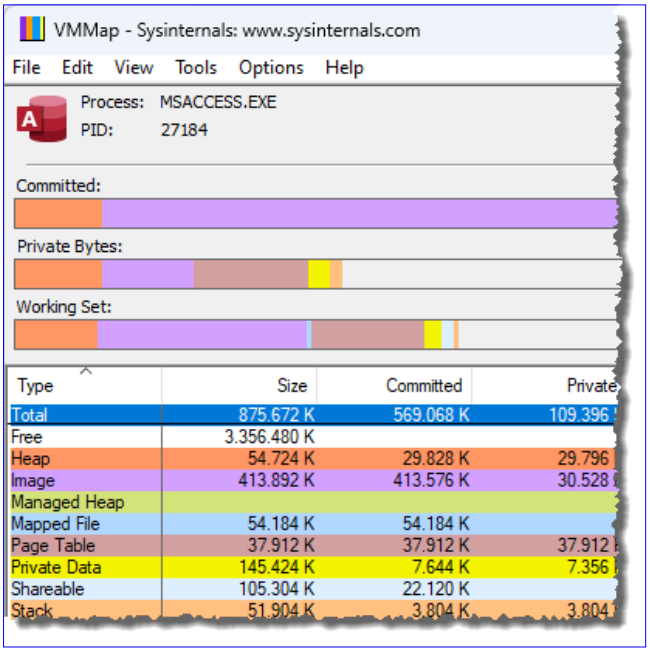

Bild 4: Speicherbedarf von Access mit aktiviertem LAA-Modus

Dekompilieren leicht gemacht

Manchmal treten in Access-Datenbanken nicht erklärbare Fehler auf. Die Datenbank stürzt immer an der gleichen Stelle ab, scheinbar ohne dass es eine Ursache dafür gibt. Oder man hat einen Haltepunkt gesetzt und wieder entfernt und der Code wird aber immer noch an der Stelle des Haltepunkts angehalten. Solche und andere Fehler können wir mit der Komprimieren und Reparieren-Funktion nicht beheben, auch wenn der Name dies suggeriert. Es ist vielmehr noch ein kleiner zusätzlicher Schritt notwendig, nämlich das Dekompilieren des Codes. Dazu gibt es eine Befehlszeilenoption mit dem Namen /decompile. Dieser Beitrag zeigt, wie Sie eine Datenbank, die Probleme macht, gegebenenfalls selbst retten können.

Probleme in Access-Datenbanken

In Microsoft Access-Datenbanken können verschiedene Probleme auftreten, für die es scheinbar keine Erklärung gibt:

- Bei Haltepunkten wird der Code auch nach dem Entfernen der Haltepunkte immer wieder angehalten.
- Codezeilen werden nicht ausgeführt.
- Formulare werden nicht geöffnet oder Access stürzt direkt ab.

Ursache könnte sein, dass bei Fehlern oder Abstürzen notwendige Informationen nicht in Systemtabellen geschrieben werden oder es werden Informationen geschrieben, die nicht mehr da sind – zum Beispiel die erwähnten Geister-Haltepunkte.

Die Lösung: Dekompilieren

Um diese Probleme zu beheben, reicht das Komprimieren/Reparieren allein nicht aus. Dazu ist ein weiterer Schritt nötig, nämlich das sogenannte Dekompilieren. Access verwendet eine kompilierte Version des VBA-Codes, da dieser schneller ist als der unkompilierte. Dieser wird beim Dekompilieren komplett verworfen. Der Code wird also in seinen ursprünglichen Zustand zurückversetzt. Öffnen wir die Datenbank danach erneut, wird der Code wieder kompiliert. Fehlerhafte oder inkonsistente Objekte werden somit aus dem kompilierten Code entfernt.

Dekompilieren allein reicht aber auch nicht aus, ein anschließender Aufruf des Befehls **Datenbank komprimieren und reparieren** vervollständigt den Prozess jedoch nach unseren Erfahrungen.

Achtung: Sicherheitskopie anlegen!

Bevor wir inoffizielle Methoden wie das Dekompilieren einer Datenbank nutzen, legen wir natürlich eine Sicherheitskopie der Datenbankdatei an. Das können wir einfach im Windows Explorer durch Markieren der Datei und anschließendes Betätigen der Tastenkombinationen **Strg + C** und **Strg + V** erledigen.

Alternativ kann man dies auch von Access aus erledigen – über den Befehl **Speichern unter** im **Datei**-Menü. Hier finden wir die Option **Datenbank sichern** (siehe Bild 1). Der Vorteil dieser Methode ist, dass diese automatisch das aktuelle Datum an den Dateinamen anhängt.

Eine Datenbank dekompilieren

Das Dekompilieren kann nicht von Access aus gestartet werden. Wir können die zu dekompilierende Datenbank jedoch auf eine bestimmte Art und Weise öffnen, um dies zu erledigen. Dazu benötigen wir einen Aufruf über die

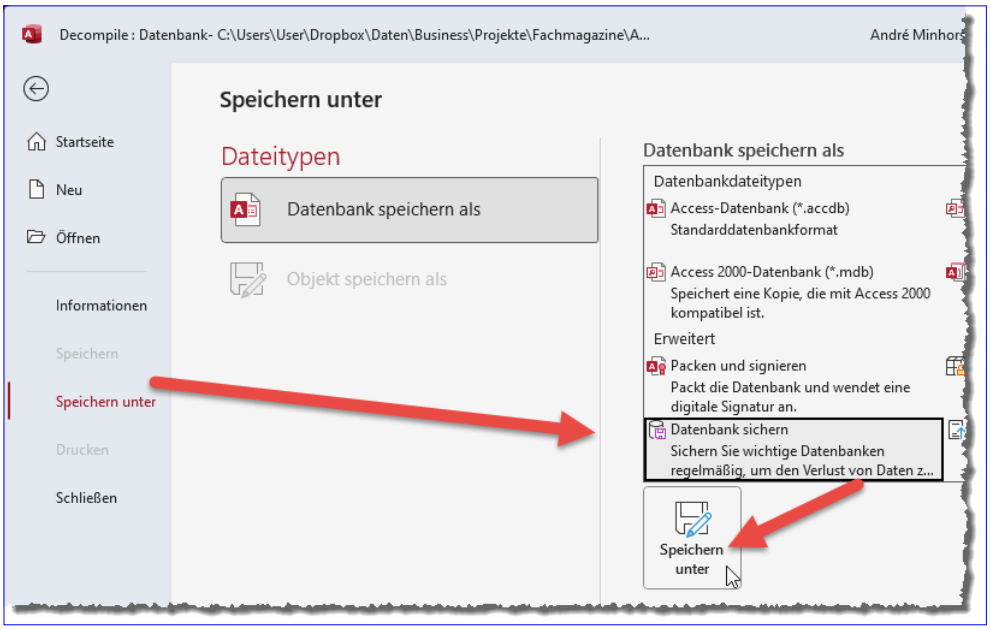

Bild 1: Sichern als Backup

Eingabeaufforderung. Diese starten wir durch Eingabe von **cmd** in das **Suchen**-Feld von Windows.

```
"C:\Program Files (x86)\Microsoft Office\root\Office16\
MSACCESS.EXE" "C:\...\Decompile.accdb" /decompile
```

Dazu geben wir als Erstes den Pfad zur Datei **MSAccess.exe** ein. Dieser lautet für Windows 11 mit Access 365 in der 32-Bit-Version beispielsweise:

```
"C:\Program Files (x86)\Microsoft Office\root\Office16\
MSACCESS.EXE"
```

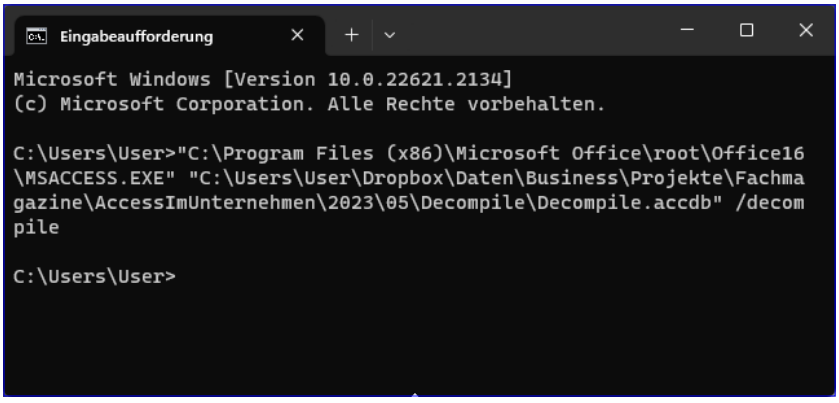

Bild 2: Aufruf über die Kommandozeile

Diesen Pfad finden wir per VBA mit dem folgenden Befehl:

```
? SysCmd(acSysCmdAccessDir)
C:\Program Files (x86)\
Microsoft Office\root\Of-
fice16\
```

Der zweite Teil ist der Pfad der zu kompilierenden Datenbankdatei. Der dritte Teil ist der Parameter namens **/decompile**.

Beim Aufruf mit diesem Befehl wie in Bild 2 wird scheinbar einfach nur die Datenbank geöffnet. Im Hintergrund wurden aber die eingangs erwähnten Schritte durchgeführt.

Hier sind allein die Ergebnisse bezüglich des Speicherplatzes der Datei auf der Festplatte:

- Vor dem Komprimieren und Reparieren: 54.016 MB
- Nach dem Komprimieren und Reparieren: 52.608 MB
- Nach dem Dekompilieren: 52.608 MB
- Nach dem erneuten Komprimieren und Reparieren: 43.136 MB

Dazu sein angemerkt, dass die meisten Tabellen in einen SQL Server ausgelagert sind – daher war kein großer Unterschied beim ersten Komprimieren und Reparieren zu bemerken. Das wir beim anschließenden Dekompilieren, Komprimieren und Reparieren fast 20% der Dateigröße verlieren, zeigt jedoch, wieviel Ballast

Outlook: E-Mail-Absender per VBA einstellen

Die üblichen Artikel über das Versenden von E-Mails über Outlook per VBA lassen meist unberücksichtigt, wie man den Absender einer E-Mail einstellen kann. Dies kann nicht durch einfaches Zuweisen der Absender-Adresse geschehen, da der Absender unter Outlook in einem Konto vorhanden sein muss. Die passende Eigenschaft heißt SendUsingAccount. Wie wir diese füllen, wie wir die verfügbaren Absender-E-Mail-Adressen ermitteln und welche Besonderheit bei Verwendung von Late Binding zu beachten sind, zeigen wir in diesem Beitrag.

Absender in Outlook auswählen

Wenn Sie in Outlook eine neue E-Mail erstellen, können Sie aus den vorhandenen Konten die gewünschte Absenderadresse auswählen (siehe Bild 1).

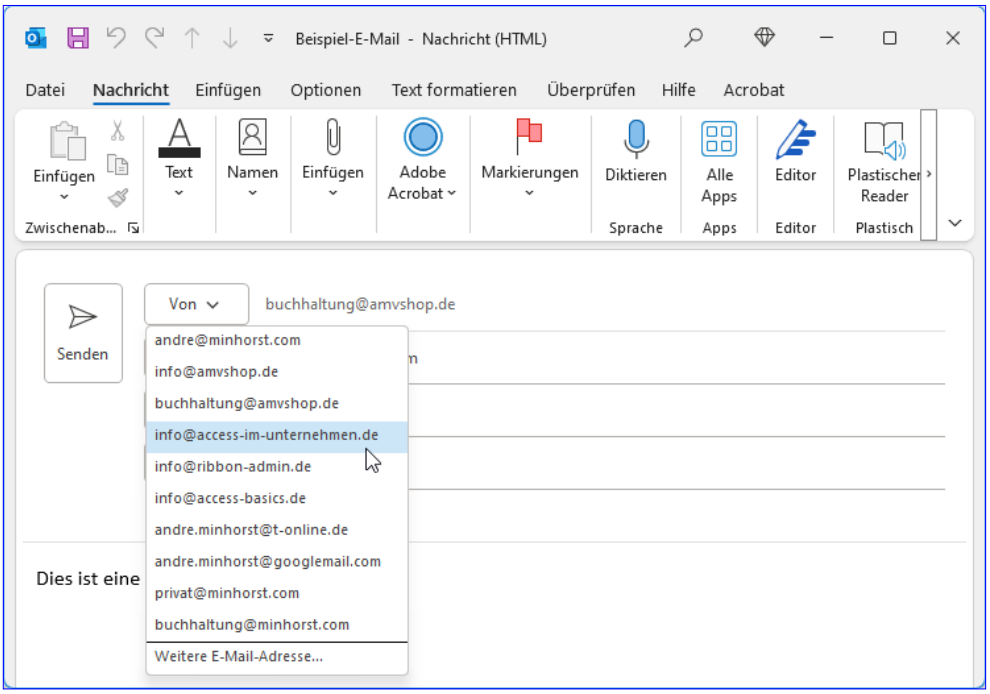

Bild 1: E-Mail-Konten, die als Absender eingestellt werden können

Wenn wir per VBA eine neue E-Mail erstellen, können wir zwar leicht den Empfänger, den Betreff, den Inhalt und andere Informationen zuweisen.

Wenn wir jedoch den Absender festlegen wollen, finden wir lediglich die Eigenschaft **SendUsingAccount** vor.

Diese nimmt jedoch keine E-Mail in Form einer Zeichenkette entgegen, sondern erwartet ein Objekt des Typs **Account**.

Also stellen wir im vorliegenden Beitrag ein Beispiel vor, mit dem Sie den gewünschten Absender auswählen und zuweisen können, sodass diese E-Mail vom entsprechenden Konto aus versendet werden kann.

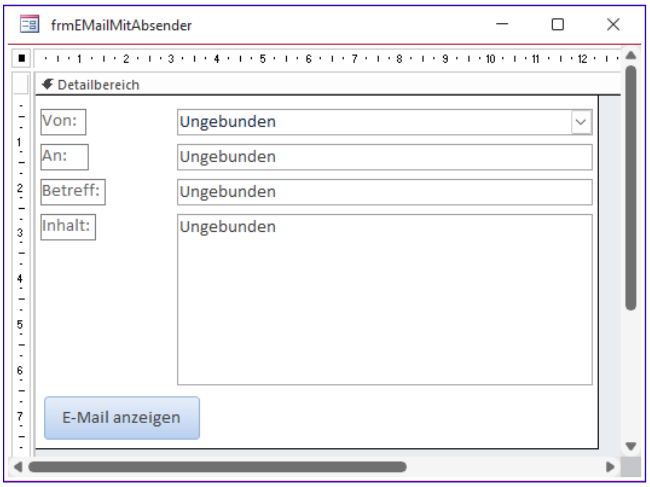

Bild 2: Formular zum Erstellen von E-Mails

```
Private Sub Form_Load()
    With Me!cboVon
        .RowSourceType = "Value List"
        .RowSource = AbsenderEinlesen
        .Value = .ItemData(0)
    End With
    Me!txtAn = Me!cboVon.ItemData(0)
    Me!txtBetreff = "Beispiel-E-Mail"
    Me!txtInhalt = "Dies ist eine Beispiel-E-Mail."
End Sub
```

Listing 1: Füllen der Formularfelder beim Öffnen des Formulars

Beispielformular

Zu diesem Zweck haben wir ein Formular erstellt, mit dem wir die gängigen Daten einer E-Mail eingeben können. Hauptsächlich wollen wir damit aber den Absender auswählen können (siehe Bild 2).

Damit uns beim Ausprobieren immer gleich ein paar Daten vorliegen, füllen wir die Felder beim Laden des Formulars mit der Prozedur aus Listing 1. Hier stellen wir insbesondere für das Kombinationsfeld zur Auswahl des Absenders die Eigenschaft **Herkunftsart** (**RowSourceType**) auf **Wertliste** ein (**Value List**).

Außerdem weisen wir der Eigenschaft **Datensatzherkunft** das Ergebnis einer Funktion namens **AbsenderEinlesen** zu und wählen direkt den ersten Eintrag als Standardwert aus.

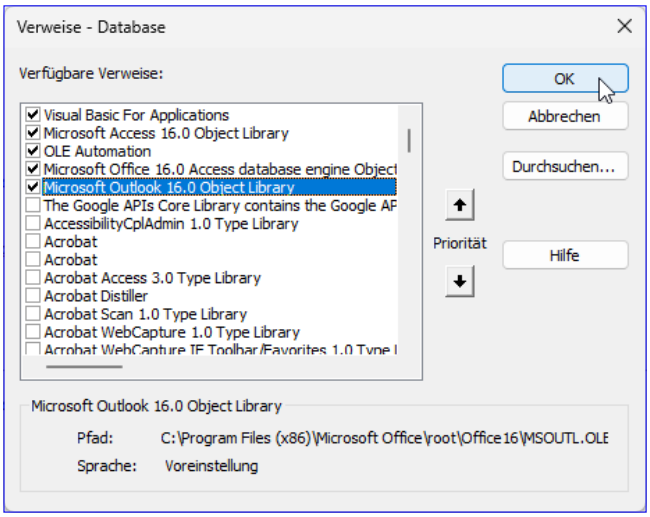

Bild 3: Verweis auf die Outlook-Objektbibliothek

Outlook per VBA und Early Binding programmieren

Damit wir beim Programmieren der Outlook-Elemente die Vorteile von Early Binding wie IntelliSense nutzen können, fügen wir dem VBA-Projekt über den **Verweise**-Dialog (**Extras|Verweise**) einen Verweis auf die Bibliothek **Microsoft Outlook x.0 Object Library** hinzu (siehe Bild 3).

Absender aus Outlook einlesen

Outlook bietet mit **Accounts** eine Auflistung an, die alle Konten der lokalen Outlook-Instanz enthält – unabhängig davon, ob diese in einer **.pst**-Datei liegen oder auf mehrere verteilt sind. Die **Accounts**-Auflistung durchlaufen

```
Public Function AbsenderEinlesen() As String
    Dim objOutlook As Outlook.Application
    Dim objMAPI As Outlook.NameSpace
    Dim objAccount As Outlook.Account
    Dim strAccounts As String
    Set objOutlook = New Outlook.Application
    Set objMAPI = objOutlook.GetNamespace("MAPI")
    For Each objAccount In objMAPI.Accounts
        strAccounts = strAccounts & objAccount.DisplayName & ";"
    Next objAccount
    AbsenderEinlesen = strAccounts
End Function
```

Listing 2: Einlesen einer Liste der möglichen Absenderadressen