Ausgabe 04/2023

ACCESS BASICS

DAS ACCESS-MAGAZIN FÜR ALLE,
DIE VON 0 AUF 100 WOLLEN

In diesem Heft:

André Minhorst

ACCESS BASICS

DAS ACCESS-MAGAZIN FÜR ALLE, DIE VON 0 AUF 100 WOLLEN

Ausgabe 04/2023

Terminplanung mit Access auf einem neuen Level

In unserem Shop gibt es ein tolles, neues Produkt: den **amvCalendar**!

Diese Lösung kannst Du problemlos in alle Access-Datenbanken integrieren. Du kannst damit Termine anlegen und diese in Kalendern in verschiedenen Ansichten wie Tag, Arbeitswoche, Woche oder Monat darstellen.

Dazu sind keine externen ActiveX-Komponenten oder DLLs nötig. Du brauchst nur die Objekte der Lösung in Deine Datenbank zu importieren und kannst den Kalender direkt anzeigen und damit arbeiten.

Der Kalender hat eine für die Anzahl der angezeigten Steuerelemente nicht für möglich zu haltende Performance.

Überzeuge Dich selbst und hole Dir die Demo von unserer Seite unter folgendem Link:

https://www.amvcalendar.de

Viel Spaß beim Ausprobieren!

André Minhorst

IMPRESSUM

ACCESS [BASICS] WIRD HERAUSGEGEBEN VON:
Minhorst und Minhorst GbR - André Minhorst Verlag
Borkhofer Straße 17
47137 Duisburg

REDAKTION:
André Minhorst (V.i.S.d.P)
Telefon: 0203/4495577
E-Mail: info@access-basics.de
Internet: www.access-basics.de

Geschäftsführung, Herstellung, Text- und Schlussredaktion, Layout von Magazin und Webseite: André Minhorst
Autor: André Minhorst
Fach- und Sprachlektorat: Carsten Gromberg

ISSN: 2190-8761

Abfragen [basics]: Parameterabfragen

Abfragen sind unter Access das Schweizer Taschenmesser zum Abfragen von Daten aus verschiedenen Feldern unterschiedlicher Tabellen mit bestimmten Kriterien, Sortierungen und Gruppierungen. Bei den Kriterien gibt es allerdings einen kleinen Haken: Gelegentlich sollen nicht feste Kriterien verwendet werden, sondern der Benutzer möchte die Kriterien zur Laufzeit eingeben können. Das ist möglich, indem er immer wieder in die Entwurfsansicht wechselt, die Vergleichswerte anpasst und wieder zur Datenblattansicht umschaltet. Allerdings macht dies auf Dauer keinen Spaß. Da passt es doch gut, dass Access mit den sogenannten Parameterabfragen die Möglichkeit bietet, die Vergleichswerte

Beispieldatenbank

Die Beispiele dieses Artikels findest Du in der Datenbank **AbfragenBasics_Parameterabfragen.accdb**.

Beispieltabelle

Wir verwenden die Tabelle **tblAutoren** als Beispieltabelle. Sie enthält verschiedene Felder wie **Vorname**, **Nachname**, **Geburtsdatum** oder **Nationalität**, mit denen wir experimentieren können.

Diese Tabelle sieht wie in Bild 1 aus.

tblAutoren

Autor-ID	Vorname	Nachname	Geburtsd	Nationalität	Zum
613	Hermann	Hesse	02.07.1877	Deutsch	
614	Jane	Austen	16.12.1775	Englisch	
615	George	Orwell	25.06.1903	Englisch	
616	Emily	Brontë	30.07.1818	Englisch	
617	William	Shakespeare	23.04.1564	Englisch	
618	Charlotte	Brontë	21.04.1816	Englisch	
619	Franz	Kafka	03.07.1883	Deutsch	
620	Mark	Twain	30.11.1835	Englisch	
621	Virginia	Woolf	25.01.1882	Englisch	
622	Ernest	Hemingway	21.07.1899	Amerikanisch	
623	Agatha	Christie	15.09.1890	Englisch	
624	Oscar	Wilde	16.10.1854	Englisch	
625	Jules	Verne	08.02.1828	Französisch	
626	Leo	Tolstoy	09.09.1828	Russisch	
627	Edgar	Poe	19.01.1809	Amerikanisch	

Datensatz: 1 von 165 | Kein Filter | Suchen

Bild 1: Tabelle mit Beispieldaten

Vorbereitung: Kriterien in einfachen Abfragen

Wenn wir eine einfache Abfrage mit einem Kriterium nutzen wollen, gehen wir beispielsweise wie folgt vor:

- Wir legen mit dem Ribbon-Befehl **Erstellen|Abfragen|Abfrageentwurf** eine neue Abfrage an.
- Dieser fügen wir die als Datensatzquelle zu verwendende Tabelle hinzu, hier **tblAutoren**.
- Dann ziehen wir alle Felder, die angezeigt oder als Kriterium verwendet werden sollen, zum Entwurfsraster der Abfrage hinzu.
- Wenn wir beispielsweise alle Autoren ermitteln wollen, deren Vorname mit **A** beginnt, fügen wir für das Feld **Vorname** das folgende Kriterium hinzu: **Wie 'A*'**

Der Abfrageentwurf sieht anschließend wie in Bild 2 aus.

Wechseln wir nun in die Datenblattansicht, erhalten wir das Ergebnis aus Bild 3.

Wollen wir nun nicht alle Autoren liefern, deren Vorname mit **A** beginnt, sondern jene, deren Vorname mit

B beginnt, müssen wir zurück in die Entwurfsansicht wechseln. Hier ändern wir den Vergleichswert für das Feld Vorname in den Ausdruck **Wie 'B*'**.

Wechseln wir anschließend wieder zur Datenblattansicht, zeigt die Abfrage alle Autoren an, deren Vorname mit **B** beginnt.

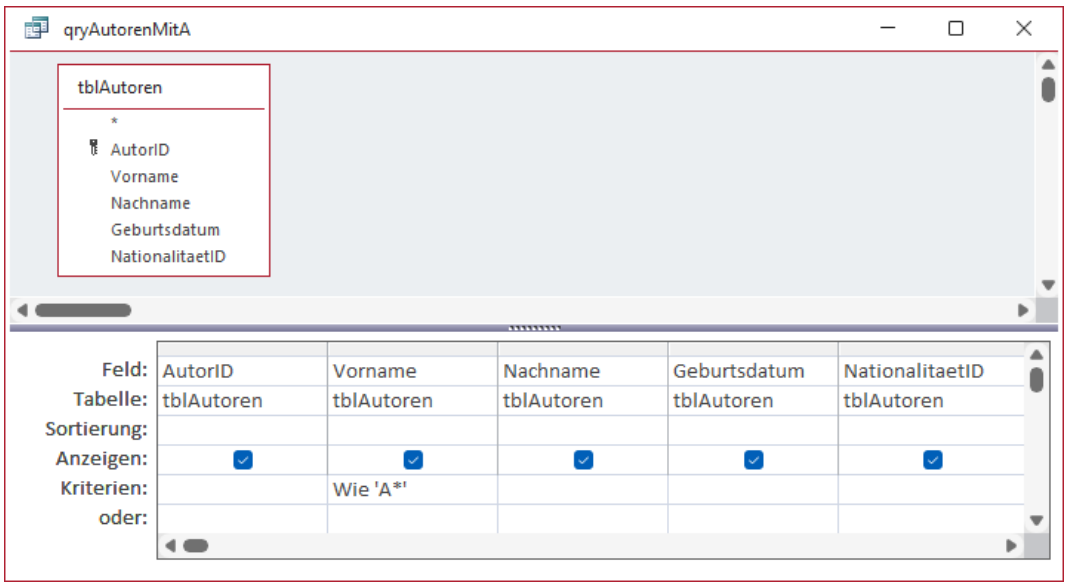

Bild 2: Abfrage zum Ermitteln aller Autoren, deren Vorname mit A beginnt.

Das kann man bestenfalls dem Entwickler einer Datenbank zumuten oder sich selbst, wenn man gelegentlich bestimmte Abfragen durchführen möchte.

Wenn wir jedoch eine Datenbank für einen Mitarbeiter oder einen Kunden entwerfen, sollten wir diesem nicht den Entwurf von Abfragen überlassen. Dafür bietet Access elegantere Möglichkeiten.

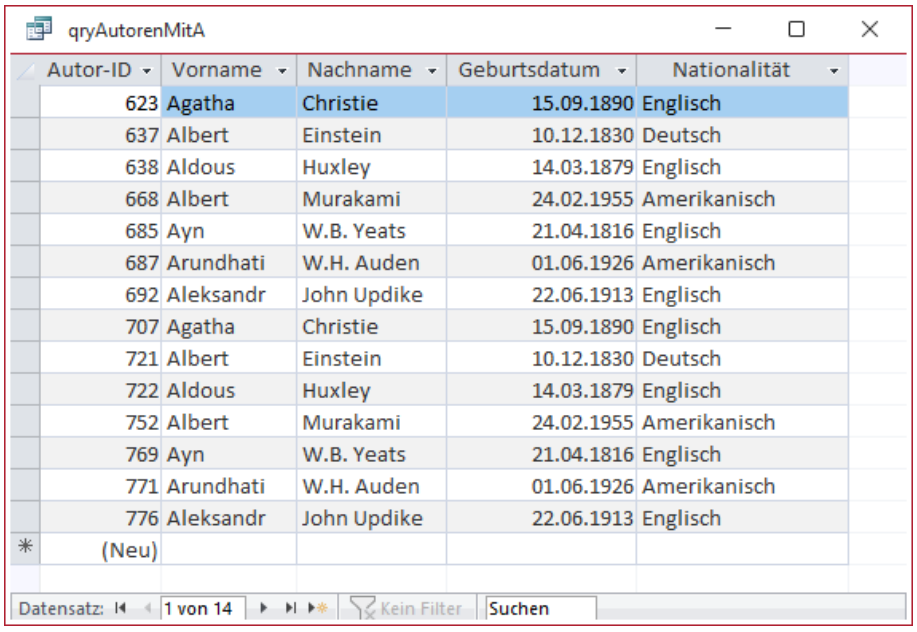

Bild 3: Abfrageergebnis mit allen Autoren, deren Vorname mit A beginnt.

Abfrage mit Parameter ausstatten

Diese Möglichkeit nennt sich Parameterabfrage. Eine solche Abfrage ist eine normale Abfrage, bei der wir für die Felder, nach denen gefiltert werden soll, Kriterien auf eine bestimmte Art festlegen.

Innerhalb dieser Kriterien können wir in eckige Klammern eingefasste Parameter definieren, deren Wert erst beim Anzeigen der Datenblattansicht der Abfrage vom Benutzer eingegeben werden muss. Wir zeigen das an einem ersten Beispiel, in welchem wir alle Datensätze ermitteln wollen, die im Feld **Vorname** einen bestimmten Wert enthalten.

Dazu erstellen wir eine Kopie der zuvor verwendeten Abfrage **qryAutorenMitA** und speichern diese unter

dem Namen **qryAutoren-NachNachname**.

In dieser ersetzen wir das Kriterium für das Feld **Vorname** durch den folgenden Ausdruck:

```
[Welcher Vorname?]
```

Der Abfrageentwurf sieht nun wie in Bild 4 aus.

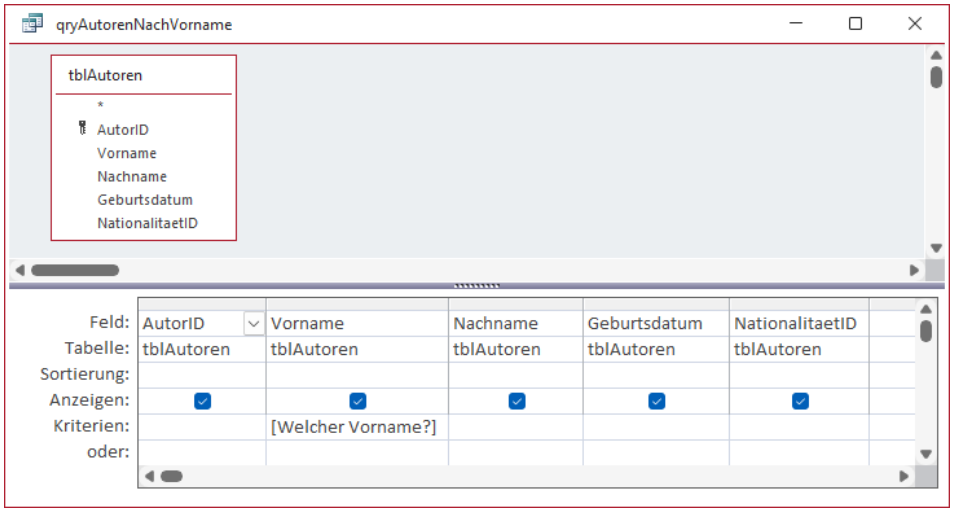

Bild 4: Abfrage mit einem Parameter für das Feld **Vorname**

Damit haben wir bereits die wichtigsten Elemente eines Parameters für ein Abfragefeld vorgestellt:

- Er wird in eckige Klammern eingefasst.
- Er enthält den Text, der später beim Abfragen des Parameterwertes angezeigt wird.

Ergebnis einer Abfrage mit Parameter anzeigen

Um diesen Parameter zu nutzen, wechseln wir einfach in die Datenblattansicht. Diese wird jedoch nicht direkt angezeigt. Zuvor erscheint eine Inputbox mit dem Titel **Parameterwert eingeben** und dem Text, den wir für den Parameter in eckigen Klammern festgelegt haben (siehe Bild 5).

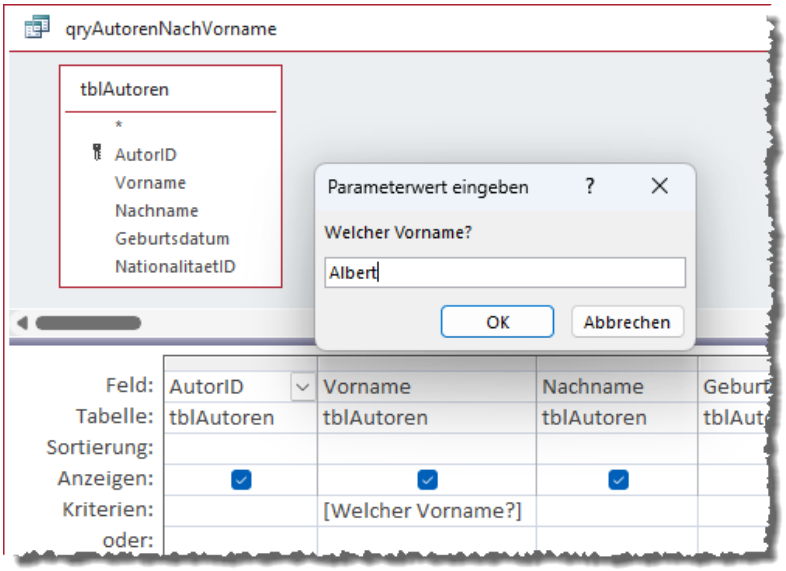

Bild 5: Eingeben des Parameterwertes

Anschließend finden wir das korrekte Ergebnis wie in Bild 6 vor.

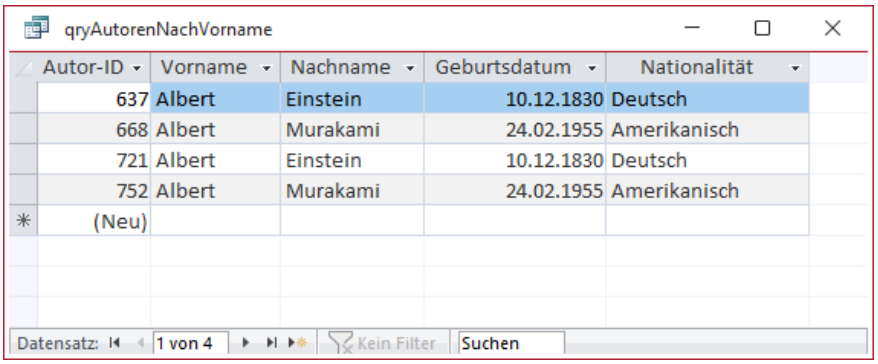

Bild 6: Abfrageergebnis mit dem Vornamen **Albert**

Regeln für die Texte von Parametern

Weiter oben haben wir einfach eine Frage in eckige Klammern eingefasst, damit diese bei der

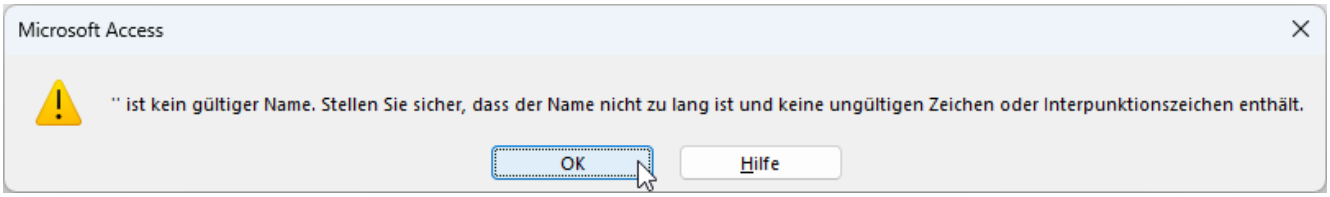

Bild 9: Fehler bei Eingabe von Punkt (.) oder Ausrufezeichen (!) im Parametertext

Anzeige der Inputbox zur Abfrage des Parameters angezeigt wird. Man könnte auch auf die Idee kommen, zur Abfrage des Vergleichswertes für das Feld **Vorname** einfach den Text **Vorname** in eckige Klammern zu schreiben, also **[Vorname]**.

Einen Feldnamen in eckigen Klammern erkennt Access jedoch nicht als Parameter, sondern als Feldname. Access verwendet dann für jeden Datensatz den Wert dieses Feldes als Vergleichskriterium.

Das heißt, Access entscheidet, ob der aktuelle Datensatz dem Kriterium entspricht, in dem es den Wert des Feldes mit dem Wert des Feldes aus dem Vergleichskriterium vergleicht – und dies liefert immer **True**. Also werden einfach alle Datensätze angezeigt. Verwende also einfach keinen der Feldnamen als Text des Parameters.

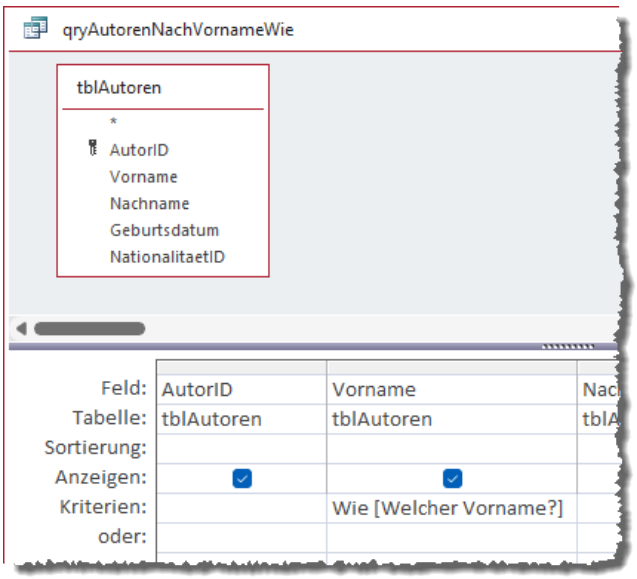

Bild 7: Abfragekriterium für Vergleichswerte mit Platzhaltern

Nicht erlaubt sind die Zeichen Punkt (.) und Ausrufezeichen (!). Verwenden wir diese, erhalten wir beim Öffnen der Abfrage eine Meldung wie in Bild 9.

Parameter mit Platzhaltern

Damit können wir nur nach Datensätzen suchen, die exakt den für den Parameter angegebenen Vornamen enthalten. Wollen wir wie in der zu Beginn vorgestellten Abfrage auch Datensätzen finden, deren Vorname mit **A** beginnt oder sogar nach solchen, die eine bestimmte Zeichenkette enthalten, müssen wir eine Änderung am Kriterium vornehmen.

Dazu gibt es verschiedene Möglichkeiten. Die wichtigste ist, dass wir das **Wie**-Schlüsselwort verwenden, statt keinen Vergleichsoperator anzugeben. Wenn wir keinen Vergleichsoperator angeben, ist das gleichbedeutend mit dem Gleichheitszeichen. Damit finden wir nur Datensätze, bei denen der Inhalt des Textfeldes exakt mit dem Vergleichswert übereinstimmt.

Die zweite Möglichkeit ist, dass wir entweder selbst Platzhalter wie das Sternchen für den Vergleichswert vorgeben oder dem Benutzer diese Aufgabe überlassen. Für eine flexiblere Abfrage wählen wir die zweite Variante. Das heißt, dass wir den Parameter wie in Bild 7 gestalten.

Wechseln wir nun in die Datenblattansicht, finden wir die gleiche Parameterabfrage vor wie zuvor, allerdings können wir diesmal auch Vergleichswerte mit Platzhaltern wie dem Stern-

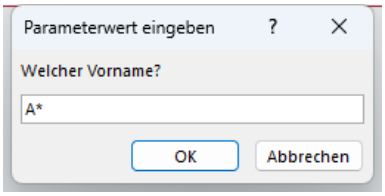

Bild 8: Platzhalter mit Sternchen eingeben

chen eingeben, also zum Beispiel wie im allerersten Beispiel **A*** (siehe Bild 8). Damit erhalten wir dann wieder alle Datensätze, deren Wert im Feld **Vorname** mit **A** beginnt.

Parameterabfrage aus Sicht des Benutzers

Wichtig bei der Parameterabfrage ist natürlich, dass der Benutzer diese einfach bedienen kann. In den Beispielen wechselten wir noch zwischen Entwurfsansicht und Datenblattansicht hin und her und konnten so die Parameterwerte erneut eingeben.

Der Benutzer soll die Entwurfsansicht eigentlich gar nicht zu sehen bekommen. Genau genommen wird er in einer professionellen Anwendung noch nicht Mal Kontakt mit Abfragen haben, sondern deren Ergebnisse in einem Formular präsentiert bekommen, über das er auch die für die Parameterwerte zu verwenden Kriterien eingeben wird.

Wenn der Benutzer jedoch selbst Parameterabfragen öffnet, zeigt ihm Access das Parameterfenster vor dem Öffnen des Abfrageergebnisses an, sodass er hier die gewünschten Vergleichswerte eingeben kann.

Erneute Parameterabfrage per Tastenkombination

Weiter oben haben wir erwähnt, dass es ebenfalls umständlich ist, bei Verwendung fester Kriterien immer wieder in den Entwurf zu wechseln und die Vergleichswerte dort anzupassen, bevor man das neue Ergebnis in der Datenblattansicht erhält. Genauso könnte man nun annehmen, der Benutzer müsse die Abfrage immer wieder schließen und erneut öffnen, um die Vergleichswerte erneut einzugeben. Das ist jedoch nicht der Fall.

Der Benutzer braucht lediglich die Taste **F5** zu betätigen, um erneut die Inputbox für den Parameterwert präsentiert zu bekommen. Diese erscheint dann über der Abfrage mit dem vorherigen Abfrageergebnis (siehe Bild 10). Nach der Eingabe eines anderen Vergleichswertes zeigt die Datenblattansicht das passende Ergebnis an.

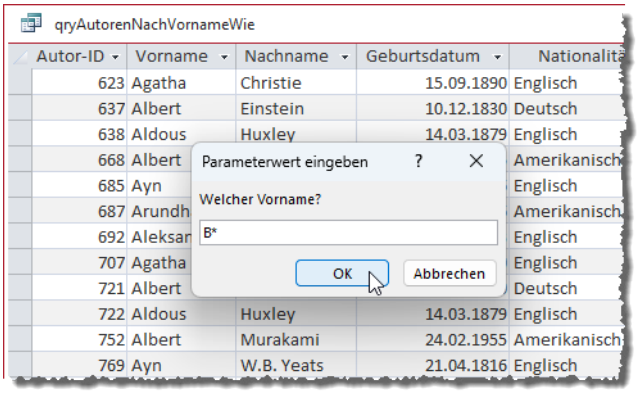

Bild 10: Erneute Parametereingabe per **F5**

Bei Textfeldern können wir zum Beispiel nach Werten suchen, welche mit der als Kriterium angegebene Zeichenkette beginnen. Dazu geben wir wie im Beispiel das Sternchen am Ende an, zum Beispiel:

```
Alb*
```

Wir können auch Datensätze suchen, deren Wert im angegebenen Feld mit einer bestimmten Zeichenkette endet:

```
*ert
```

Oder wir suchen einfach nach Datensätzen, die in diesem Feld an beliebiger Stelle das angegebene Kriterium enthalten. Das gelingt wie folgt:

```
*ber*
```

Dem Benutzer die Arbeit mit den Platzhaltern abnehmen

Gegebenenfalls möchtest Du dem Benutzer nicht aufbürden, mit Sternchen und Fragezeichen hantieren zu müssen, sondern willst ihm einfach die Möglichkeit geben, alle Vornamen zu finden, die eine bestimmte Zeichenkette enthalten.

Dann kannst Du den Parameter in die entsprechenden Platzhalter einbetten wie in folgendem Ausdruck:

```
Wie "*" & [Vorname soll enthalten:] & "*"
```

Eingebaute Access-Funktionen nutzen

Access bietet eine ganze Reihe eingebauter Funktionen an, die wir an verschiedenen Stellen einer Datenbankanwendung nutzen können – in den Eigenschaften von Tabellen, Abfragen, Formularen, Berichten und Makros. Und für die meisten gibt es ein VBA-Pendant, mit dem wir diese Funktionen auch im Code verwenden können. Dieser Artikel zeigt anhand einiger Beispiele, wie wir solche Funktionen nutzen können und wie wir diese überhaupt finden.

Beispieldatenbank

Die Beispiele dieses Artikels findest Du in der Datenbank **EingebauteFunktionenNutzen.accdb**.

Anwendungsbereiche eingebauter Funktionen

Die Anwendungsbereiche eingebauter Funktionen erstrecken sich über alle Objekttypen von Access:

- Eigenschaften in Tabellen
- Eigenschaften von Tabellenfeldern
- Eigenschaften von Abfragen
- Eigenschaften von Abfragefeldern
- Werte von Abfragefeldern
- Kriterien für Abfragefelder
- Eigenschaften von Formularen
- Eigenschaften von Berichten
- Eigenschaften von Steuerelementen
- Eigenschaften beziehungsweise Parameter von Makroaktionen

Eingebaute Funktionen in Tabellen

In den Eigenschaften einer Tabelle bietet sich nur das Feld **Gültigkeitsregel** als Einsatzort für eingebaute Funktionen an. Hier können wir zum Beispiel prüfen, ob Bedingungen erfüllt sind, die mehrere Felder betreffen.

Wenn wir beispielsweise sicherstellen wollen, dass mindestens für eines von zwei Feldern ein Wert eingegeben wird, zum Beispiel für die Felder **Telefon** und **E-Mail**, können wir die Funktion **IsNull** nutzen.

In diesem Fall weisen wir der Eigenschaft **Gültigkeitsregel** den folgenden Ausdruck zu:

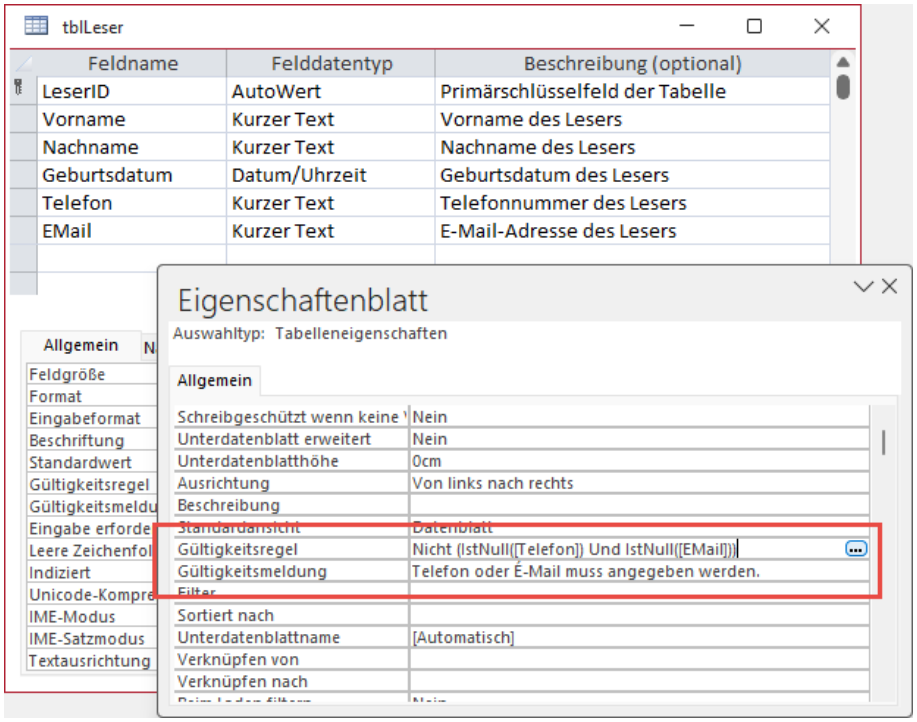

Bild 1: Beispiel für eine Gültigkeitsregel mit eingebauten Funktionen

```
Nicht (IstNull([Telefon]) Und
IstNull([EMail]))
```

Damit lautet die Gültigkeitsregel übersetzt, dass nicht beide Felder den Wert **Null** enthalten dürfen, also leer sind. Anderenfalls wird die für die Eigenschaft **Gültigkeitsmeldung** angegebene Meldung (siehe Bild 1) beim Versuch, den Datensatz zu speichern, ausgegeben.

Eingebaute Funktionen in Tabellenfeldern

In den Eigenschaften von Tabellenfeldern finden wir weitere Möglichkeiten für den Einsatz eingebauter Funktionen. Neben der Eigenschaft **Gültigkeitsmeldung** ist dies hier auch für die Eigenschaft **Standardwert** möglich.

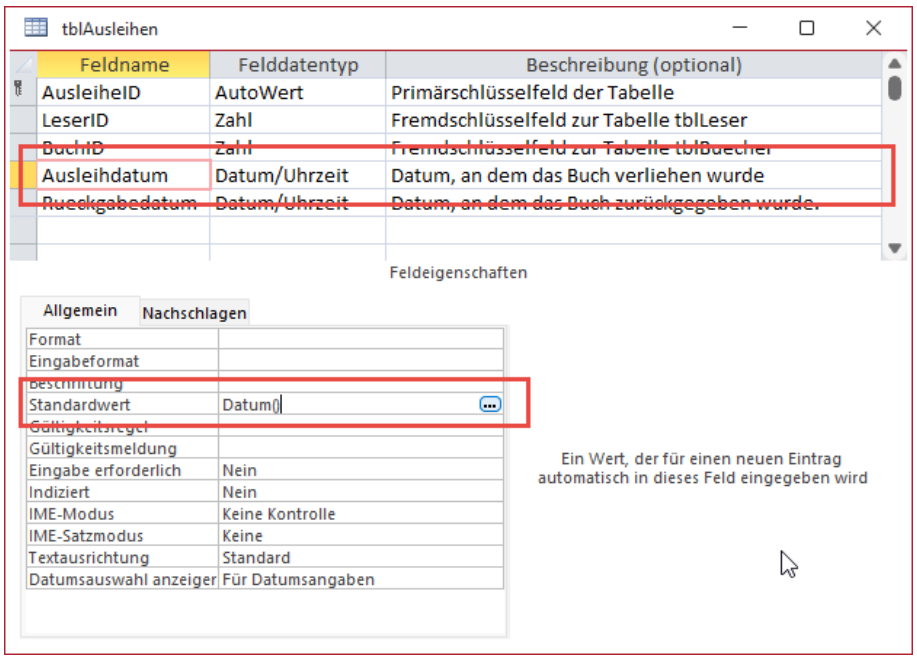

Bild 2: Eingebaute Funktionen in den Feldeigenschaften einer Tabelle

In Bild 2 sehen wir einen Klassiker – das Festlegen des aktuellen Datums als Standardwert für ein Datumsfeld mit der Funktion **Datum()**. Damit wird das Feld **AngelegtAm** immer direkt beim Anlegen eines neuen Datensatzes mit dem aktuellen Datum gefüllt.

Das können wir noch erweitern, indem wir die zusätzlich die Funktion **Jetzt()** verwenden:

```
Datum() + Jetzt()
```

Wir verwenden hier das Plus-Zeichen (**+**) und nicht etwa das Kaufmanns-Und (**&**), da **Datum()** und **Jetzt()** intern Zahlenwerte liefern, die addiert Datum und Uhrzeit ergeben.

Eingebaute Funktionen in Abfragen

Die Eigenschaften von Abfragen selbst bieten keine Gelegenheit, eingebaute Funktionen anzuwenden. Jedoch gibt es einige Möglichkeiten in den Eigenschaften der Felder im Abfrageentwurf und auch für den Feldwert und die Kriterien können wir eingebaute Funktionen einsetzen.

Eingebaute Funktionen als Kriterien von Abfragefeldern

Der erste Einsatzzweck von eingebauten Funktionen dürften die Kriterien der Felder einer Abfrage sein. Natürlich können wir hier feste Werte eingeben oder auch Parameter nutzen, wie wir es im Artikel **Abfragen [basics]: Parameterabfragen** (**www.access-basics.de/605**) beschreiben.

Aber wir können auch dynamisch Kriterien zusammensetzen und so Daten aus anderen Tabellen, das aktuelle Datum und andere Informationen als Kriterium verwenden.

Die im Artikel **Eingebaute Funktionen: Domänenfunktionen** (**www.access-basics.de/630**) vorgestellten Domänenfunktionen können wir beispielsweise als Kriterium für Abfragefelder nutzen.

Aber es gibt wesentlich einfachere Funktionen, die wir in Abfragen einsetzen können. Wenn wir zum Bei-

Domänenfunktionen: Vergleichskriterien nutzen

Wer mit Domänenfunktionen arbeitet und dabei Kriterien nutzen möchte, kommt um grundlegende Kenntnisse der **WHERE**-Klausel von Access-SQL nicht aus. Der dritte Parameter der Domänenfunktionen wie **DLookup**, **DCount** oder **DMax** folgt nämlich den gleichen Regeln wie die **WHERE**-Klausel. Welche dies sind, besprechen wir in diesem Artikel – natürlich nicht ohne ausreichend praktische Beispiele. Dabei schauen wir uns einfache Vergleiche von zwei Werten wie beispielsweise einem Feldwert und einem Vergleichswert an.

Beispieldatenbank

Die Beispiele dieses Artikels findest Du in der Datenbank **Domaenenfunktionen_KriterienZusammenstellen.accdb**.

Kriterien ausprobieren

Zum Ausprobieren der Kriterien für eine Domänenfunktion haben wir auch der Beispieldatenbank dieses Artikels das Formular **frmDomaenenfunktionen** beigefügt. Wenn Du den ersten und zweiten Parameter definiert hast, kannst Du die Beispiele aus dem vorliegenden Artikel für den dritten Parameter Kriterien einfügen. Die Anzeige der Ergebnisse für die verschiedenen Domänenfunktionen wird nach der Eingabe eines jeden Zeichens aktualisiert (siehe Bild 1).

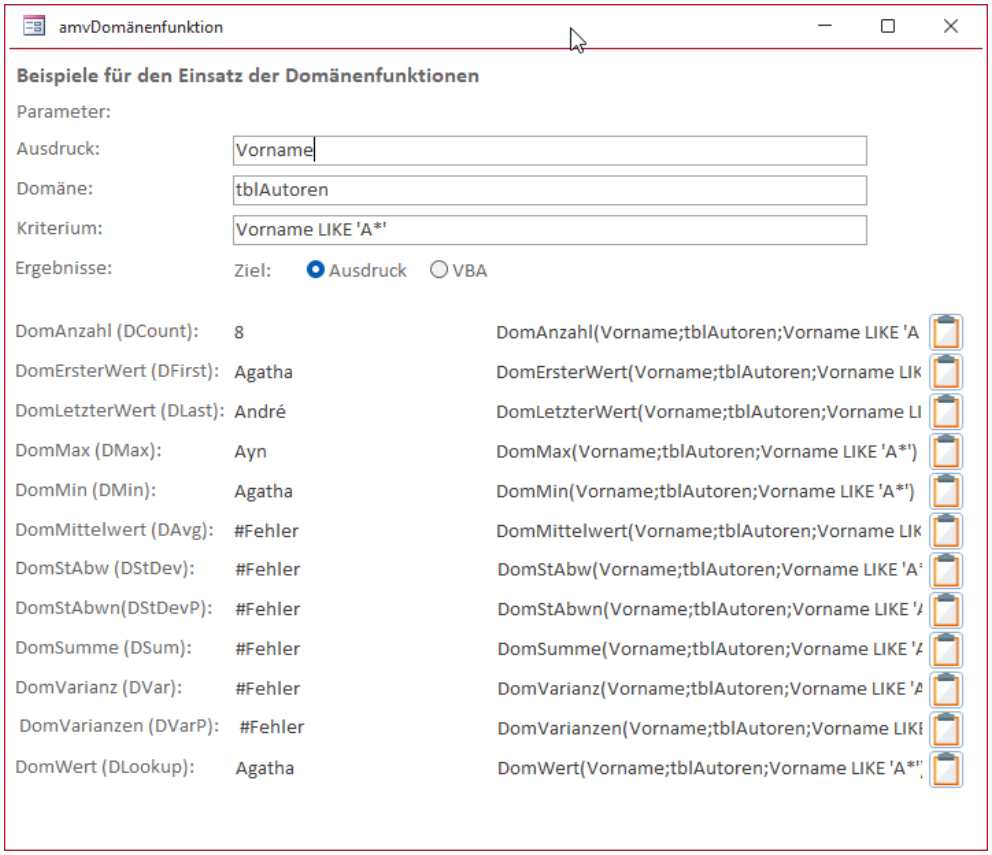

Bild 1: Testen der Kriterien für Domänenfunktionen

Aufbau von Kriterien

Im Rahmen des vorliegenden Artikels schauen wir uns die Definition von Vergleichswerten an. Dabei vergleichen wir immer zwei Ausdrücke miteinander. Die Ausdrücke können dabei beliebig gestaltet sein – wir können die Inhalte zweier Felder miteinander vergleichen, den Inhalt eines Feldes mit einem fest vorgegebene Wert oder den Inhalt eines Feldes mit dem Ergebnis einer Funktion.

Wichtig ist, dass das Kriterium bei allen folgenden Beispielen aus drei Elementen aufgebaut ist: dem ersten Wert, dem Vergleichskriterium und dem

zweiten Wert. Es gibt noch weitere Möglichkeiten, Kriterien für Domänenfunktionen und auch für den **WHERE**-Teil von SQL-Abfragen zu definieren – zum Beispiel die Verwendung von Funktionen oder die Verknüpfung verschiedener Kriterien mit **AND** oder **OR**. Auf beide gehen wir weiter unten ein.

Eckige Klammern für Feldnamen

Nicht verpflichtend, aber in manchen Fällen notwendig ist das Einfassen von Feldnamen in Kriteriumsausdrücken in eckige Klammern.

Wenn wir immer nur die standardmäßig vorgesehenen Zeichen für Feldnamen verwenden, brauchen wir diese nicht, aber wenn man Zeichen wie Leerzeichen, Minuszeichen et cetera verwendet, sollte man diese in Kriteriumsausdrücken in eckige Klammern einfassen:

```
[Sonder-Zeichen] = ""
```

Wenn wir das wie folgt nicht tun, interpretiert Access das Minus-Zeichen als Rechenzeichen, was zu einem Fehler führt:

```
Sonder-Zeichen = ""
```

Vergleichsoperatoren für Zahlenwerte

Wenn wir im Kriterium zwei Zahlenwerte vergleichen wollen, stehen uns verschiedene Vergleichsoperatoren zur Verfügung:

- Gleich (**=**): Liefert die Datensätze, bei denen der Feldwert gleich dem Vergleichswert ist (zum Beispiel **[Feldname] = 1**).
- Ungleich (<>): Liefert die Datensätze, bei denen der Feldwert ungleich dem Vergleichswert ist (zum Beispiel **[Feldname] <> 1**).
- Nicht gleich (**NOT** mit **=**): Liefert die Datensätze, bei denen der Feldwert nicht gleich dem Vergleichswert ist (zum Beispiel **NOT [Feldname] = 1**).
- Größer (**>**): Liefert die Datensätze, bei denen der erste Wert größer als der zweite Wert ist (zum Beispiel **[Feldname] > 1**])
- Kleiner (**<**): Liefert die Datensätze, bei denen der erste Wert kleiner als der zweite Wert ist (zum Beispiel **[Feldname] < 10**)
- Größer oder gleich(**>=**): Liefert die Datensätze, bei denen der erste Wert größer als der zweite Wert oder gleich ist (zum Beispiel **[Feldname] >= 1**])
- Kleiner oder gleich (**<=**): Liefert die Datensätze, bei denen der erste Wert kleiner als der zweite Wert oder gleich ist (zum Beispiel **[Feldname] <= 10**)

Kriterium mit Zahlenbereich

Kriterien, welche die oben genannten Vergleichsoperatoren nutzen, können wir mit dem Schlüsselwort **AND** so verknüpfen, dass wir Zahlenbereiche erhalten. Hier ist auch noch entscheidend, ob wir die als Vergleichswert angegebene Zahl einschließen oder nicht. Um alle Zahlen zwischen **21** und **30** zu erhalten, verwenden wir das folgende Kriterium:

```
[Feldname]>=21 AND [Feldname]<=30
```

Für Ganzzahlen könnten wir auch schreiben:

```
[Feldname]>20 AND [Feldname]<31
```

Zahlenbereich ausschließen

Wenn wir einen Bereich ausschließen wollen, also beispielsweise alle Ganzzahlen ohne die Zahlen von **21** bis **30** berücksichtigen wollen, nutzen wir folgenden Ausdruck. Wichtig ist, dass wir hier nicht das **AND**-Schlüsselwort nutzen können, sondern **OR**:

```
[Feldname] <721 OR [Feldname]>730
```

Wir können auch den auszuschließenden Bereich wie oben mit **AND** definieren, das Kriterium in Klammern einfassen und das Schlüsselwort **NOT** voranstellen:

```
NOT ([Feldname]>=21 AND [Feldname]<=30)
```

Vergleiche mit Ganzzahlen und Dezimalzahlen

Ein wichtiger Unterschied ergibt sich dadurch, ob wir es mit einem Feld zum Speichern von Ganzzahlen oder von Dezimalzahlen zu tun haben. Wenn es sich um ganzzahlige Werte handelt, können wir auch nur ganzzahlige Werte als Vergleichswerte nutzen. Anderenfalls erhalten wir kein korrektes Ergebnis.

Der Workaround ist, dass wir den als Vergleichskriterium verwendeten Wert zuvor mit der **Int**-Funktion um die Nachkommastellen erleichtern:

```
[AutorID[ > Int(1.6)
```

Wenn wir ein Zahlenfeld mit Nachkommastellen verwenden, können wir nach Belieben auch Zahlen mit Nachkommastellen als Kriterium verwenden.

Hier ist nur zu beachten, dass wir den Punkt (**.**) als Dezimaltrennzeichen nutzen:

```
[Dezimalfeld] > 1.6
```

Vergleich mit Boolean-Werten (Ja/Nein)

Bei der Verwendung von **Boolean**-Werten in den Kriterien von Domänenfunktionen ist zu beachten, dass wir nicht wie im Abfrageentwurf einfach **Ja/Nein**, **Wahr/Falsch** et cetera eingeben wollen. Access-SQL verlangt nach **-1**/**True** oder **0**/**False**.

Funktonierende Ausdrücke lauten also:

```
[Aktiv] = -1
[Aktiv] = 0
[Aktiv] = TRUE
[Aktiv] = FALSE
```

TRUE und **FALSE** haben wir groß geschrieben, weil es Schlüsselwörter von Access-SQL sind.

Vergleich mit Zeichenketten

Der Vergleich mit Zeichenketten unterscheidet sich von dem mit den anderen Datentypen durch die notwendigen Anführungszeichen oder Hochkommata.

Der Einfachheit halber wollen wir zunächst mit Hochkommata arbeiten. Ein einfacher Ausdruck sieht wie folgt aus:

```
[Vorname] = 'André'
```

Mit dem Gleichheitszeichen können wir auf exakte Übereinstimmung prüfen. Beim Vergleich mit Zeichenketten stehen uns allerdings auch Platzhalter zur Verfügung:

- Sternchen (*****): Steht für kein, ein oder mehrere beliebige Zeichen.
- Fragezeichen (**?**): Steht für ein beliebiges Zeichen.

Um diese nutzen zu können, verwenden wir statt des Gleichheitszeichens den Operator **LIKE**. Um dann alle Namen zu erhalten, die mit **A** beginnen, nutzen wir:

```
[Vorname] LIKE 'A*'
```

Wenn wir alle Namen **Andre**, **André** oder **Andrè** erhalten wollen, können wir auch nur für das letzte Zeichen einen Platzhalter eingeben, in diesem Fall das Fragezeichen:

```
[Vorname] LIKE 'Andr?'
```

Größer und Kleiner bei Zeichenketten

Wir können auch alle Datensätze ermitteln, bei denen ein Textfeld in einem bestimmten Bereich liegt. Wenn wir beispielsweise alle Vornamen ermitteln wollen, die zwischen **A** und **M** liegen, verwenden wir:

```
[Vorname] >= 'A' AND [Vorname] <'N'
```

Eingebaute Funktionen: Domänenfunktionen

Access bietet eine Menge Funktionen an, die in den Eigenschaften und Ausdrücken von Access-Objekten wie Tabellen, Abfragen, Formularen, Berichten und Makros verwendet werden können. Einige davon dienen der Abfrage von Informationen aus Tabellen und Abfragen – die sogenannten Domänenfunktionen. Entsprechend beginnen die Namen dieser Funktionen mit **Dom** – wie **DomWert**, **DomMax**, **DomMin** und so weiter. In diesem Artikel zeigen wir, welche Domänenfunktionen es gibt, welche Funktion sie haben und wie Du diese in Deinen eigenen Anwendungen nutzen kannst.

Beispieldatenbank
Die Beispiele dieses Artikels findest Du in der Datenbank **EingebauteFunktionen_DLookupUndCo.accdb**.

Domänenfunktionen
In Microsoft Access werden Domänenfunktionen verwendet, um Berechnungen über bestimmten Daten zu erstellen, die in Tabellen oder Abfragen gespeichert sind. Diese Funktionen liefern aggregierte Ergebnisse, wie zum Beispiel den maximalen Wert oder die Anzahl der Datensätze, die bestimmten Kriterien entsprechen.

Hier sind die Domänenfunktionen in Access. Vorn steht immer die englische Bezeichnung, in Klammern die deutsche. Wir können grundsätzlich beide als Ausdruck angeben, in der deutschen Version werden die Bezeichnungen jedoch automatisch nach der Eingabe in die deutsche Version geändert:

- **DAvg** (**DomMittelwert**): Berechnet den Durchschnittswert eines bestimmten Felds in einer Tabelle oder Abfrage, basierend auf den angegebenen Kriterien.
- **DCount** (**DomAnzahl**): Ermittelt die Anzahl der Datensätze in einer Tabelle oder Abfrage, die den angegebenen Kriterien entsprechen.
- **DFirst** (**DomErsterWert**): Gibt den ersten Wert eines bestimmten Felds in einer Tabelle oder Abfrage zurück, basierend auf den angegebenen Kriterien.
- **DLast** (**DomLetzterWert**): Gibt den letzten Wert eines bestimmten Felds in einer Tabelle oder Abfrage zurück, basierend auf den angegebenen Kriterien.
- **DLookup** (**DomWert**): Gibt den Wert eines bestimmten Felds aus einer bestimmten Tabelle oder Abfrage zurück, basierend auf den angegebenen Kriterien.
- **DMax** (**DomMax**): Gibt den maximalen Wert eines bestimmten Felds in einer Tabelle oder Abfrage zurück, basierend auf den angegebenen Kriterien.
- **DMin** (**DomMin**): Gibt den minimalen Wert eines bestimmten Felds in einer Tabelle oder Abfrage zurück, basierend auf den angegebenen Kriterien.
- **DSum** (**DomSumme**): Berechnet die Summe eines bestimmten Felds in einer Tabelle oder Abfrage, basierend auf den angegebenen Kriterien.
- **DVar** (**DomVarianz**): Berechnet die Varianz eines bestimmten Felds in einer Tabelle oder Abfrage, basierend auf den angegebenen Kriterien.
- **DVarP** (**DomVarianzen**): Berechnet die populationsbasierte Varianz eines bestimmten Felds in einer Tabelle oder Abfrage, basierend auf den angegebenen Kriterien.
- **DStDev** (**DomStAbw**): Berechnet die Standardabweichung eines bestimmten Felds in einer Tabelle

oder Abfrage, basierend auf den angegebenen Kriterien. Die Funktion verwendet die "Stichproben"-Standardabweichung (N-1).

- **DStDevP** (**DomStAbwn**): Berechnet die populationsbasierte Standardabweichung eines bestimmten Felds in einer Tabelle oder Abfrage, basierend auf den angegebenen Kriterien. Die Funktion verwendet die "populations"-basierte Standardabweichung (N).

Diese Domänenfunktionen sind besonders nützlich, wenn du aggregierte Informationen aus deiner Datenbank abfragen möchtest und an beliebigen Stellen weiterverarbeiten oder anzeigen möchtest. Im Folgenden schauen wir uns die verschiedenen Domänenfunktionen im Detail an.

Grundsätzliche Informationen zu Domänenfunktionen

Domänenfunktionen sind eine Vereinfachung zur Erstellung einer SQL-Abfrage zum Ermitteln von Daten. Während SQL-Abfragen jedoch keinen, einen oder mehrere Datensätze zurückliefern können, enthält das Ergebnis einer Domänenfunktion immer nur die Daten eines Datensatzes. Und wir können auch nicht wie in einer SQL-Abfrage eines oder mehrere Felder zurückliefern, sondern müssen alle zu ermittelnden Daten in einem einzigen Ausdruck definieren, der dann als Ergebnis zurückgeliefert wird.

Wir können also beispielsweise nur den Primärschlüsselwert eines Datensatzes zurückliefern, der im Feld **Nachname** einen bestimmten Wert enthält oder den kleinsten auftretenden Wert im Feld **Geburtsdatum** einer Tabelle.

Die Domänenfunktionen haben jeweils drei Parameter:

- **Ausdruck**: Gibt an, welche Informationen aus der mit dem zweiten Parameter angegebenen Tabelle oder Abfrage gelieferten Datensätzen ermittelt werden sollen. Dabei kann es sich um den Inhalt eines einzelnen Feldes handeln, es können aber auch mehrere Felder zusammengefasst zurückgegeben werden. Im Ausdruck können wir sogar andere eingebaute Funktionen nutzen, um die Ausgabe anzupassen – beispielsweise um die zurückgegebene Zeichenkette zu bearbeiten.

- **Domäne**: Enthält die Angabe einer Tabelle oder Abfrage. Diese muss in der aktuellen Datenbank gespeichert sein.

- **Kriterien** (optional): Hier können wir die Menge der zurückgegebenen Datensätze einschränken. Dabei verwenden wir einen Ausdruck, der ein oder mehrere Kriterien enthält, die mit **Und** oder **Oder** verknüpft sind. Wenn wir kein Kriterium eingeben, durchsucht Access die komplette Abfrage oder Tabelle nach den von der Domänenfunktion zu liefernden Daten.

Hinweise zum Festlegen des Ausdrucks

Der erste Parameter legt fest, welchen Ausdruck die Funktion als Ergebnis berücksichtigen soll. Hier gibt es eine wichtige Unterscheidung. Manche der Domänenfunktionen liefern einfach den Wert des ersten gefundenen Datensatzes zurück, der diesem Ausdruck entspricht.

Dazu gehören **DLookup**, **DMax**, **DMin**, **DFirst** und **DLast**. Hier können wir beliebige Ausdrücke zusammenstellen, also zum Beispiel den Inhalt zweier Textfelder mit **[Vorname] & " " & [Nachname]** verbinden oder Berechnungen durchführen wie **[Einzelpreis] * [Mehrwertsteuersatz]**. Da immer nur der Wert eines dieser Ausdrücke geliefert wird, können diese beliebig gestaltet werden.

Andere liefern berechnete Werte, die sich auf alle Datensätze beziehen, die durch das Kriterium eingeschlossen werden. Dabei handelt es sich um **DAvg**, **DSum**, **DVar**, **DVarP**, **DStDev** und **DStDevP**.

Diese liefern einen berechneten Wert auf Basis des angegebenen Ausdrucks. Es ist daher wichtig, dass

der Ausdruck einen numerischen Wert liefert. Für ein Textfeld beispielsweise würden diese Funktionen den Wert **#Fehler** zurückgeben.

Schließlich gibt es noch die Funktion **DCount**, die einfach die Anzahl zurückgibt. Bei **DCount** gibt es wie bei allen anderen Funktionen nur eines zu berücksichtigen – siehe nächster Abschnitt.

Verhalten der Domänenfunktionen beim Wert Null

Wenn eine der Domänenfunktionen, die einen konkreten Wert zurückgibt, auf einen Datensatz trifft, wo dieser Wert **Null** lautet, wird einfach **Null** zurückgegeben.

Bei den Funktionen, die Datensätze zählen, Daten summieren oder eine andere der Operationen durchführen, die sich auf die Gesamtheit der durch das Kriterium angegebenen Datensätze beziehen, werden Datensätze mit dem Wert Null im zu untersuchenden Ausdruck ignoriert. Wenn wir also beispielsweise Datensätze zählen wollen und dabei ein Feld als Ausdruck angeben, das in einigen Datensätzen den Wert **Null** enthält, dann erhalten wir als Ergebnis die Anzahl der Datensätze, in denen dieses Feld gefüllt ist.

Testen der Domänenfunktionen

Wenn Du die Domänenfunktionen ausgiebig testen möchtest, kannst Du das in der Datenbank enthaltene Formular **frmDomaenenfunktionen** nutzen. Hier gibst

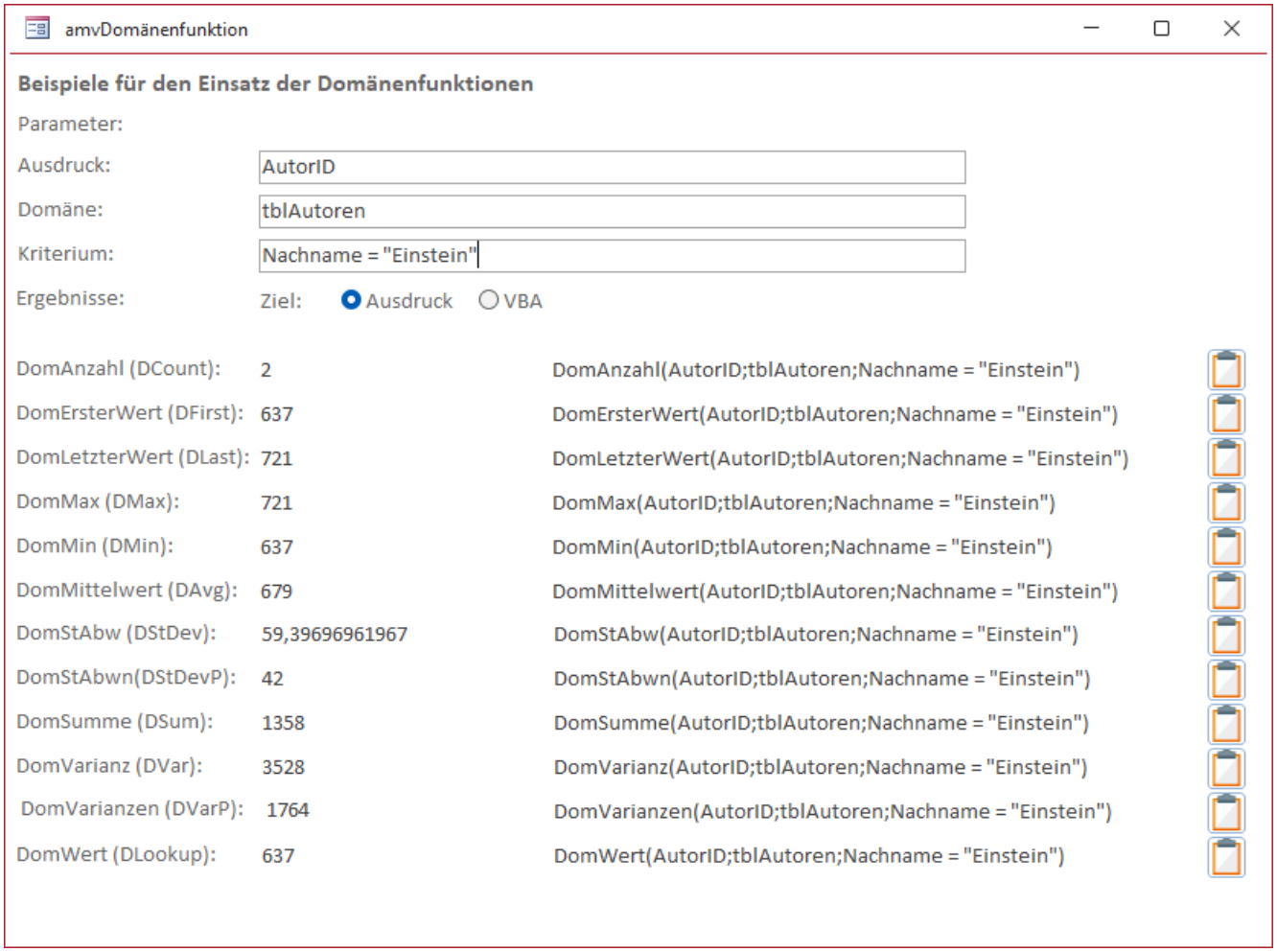

Bild 1: Formular-Entwurf unseres Beispielformulars

Formulare [basics]: Forms öffnen mit DoCmd.OpenForm

Access-Anwendungen brauchen Formulare für die Interaktion mit dem Benutzer. Aber wie öffnet man ein solches Formular überhaupt? Klar, das funktioniert zwar mit einem Doppelklick auf den Namen des jeweiligen Formulars im Navigationsbereich. Aber diesen sollten der Benutzer im Optimalfall überhaupt nicht sehen, denn sonst könnte er auch direkt auf die Inhalte von Tabellen zugreifen – und dazu soll er eigentlich die an die Tabellen gebundenen Formulare nutzen. Wir müssen also eine alternative Möglichkeit schaffen, damit der Benutzer die Formulare öffnen kann. Diese soll möglichst flexibel sein und es auch einmal erlauben, Informationen wie einen Filter zu übergeben oder ob das zu öffnende Formular einen neuen, leeren Datensatz anzeigen soll. Das Schweizer Taschenmesser für diesen Fall ist die Methode **OpenForm** der **DoCmd**-Klasse. Damit können wir Formulare in verschiedenen Ansichten öffnen, Parameter übergeben, Filter setzen oder auch den Datenbearbeitungsmodus einstellen.

Beispieldatenbank

Die Beispiele dieses Artikels findest Du in der Datenbank **FormulareBasics_FormulareOeffnenPerDoCmdOpenForm.accdb**.

Formulare öffnen

Das Öffnen von Formularen ist eine essenzielle Aktion in Access-Datenbanken, in denen der Benutzer auf irgendeine Weise mit den Daten interagieren soll. Um dies zu erledigen, gibt es die folgenden Möglichkeiten:

- Doppelklick auf den Formularnamen im Navigationsbereich
- Angabe eines Formulars für die Option **Formular anzeigen** in den Access-Optionen
- Öffnen eines Formulars mit dem Makrobefehl **ÖffnenFormular**
- Öffnen eines Formulars mit der VBA-Methode **DoCmd.OpenForm**

Doppelklick auf den Formularnamen

Die erste Methode fällt aus den in der Einleitung genannten Gründen aus – der Benutzer sollte den Navigationsbereich normalerweise gar nicht zu Gesicht bekommen.

Startformular in den Optionen festlegen

Die zweite Methode ist gut geeignet, wenn Du ein Startformular zur Anzeige eines Splash Screens beim Öffnen anzeigen möchtest oder Du ein Formular mit den Funktionen der Datenbankanwendung anzeigen möchtest, das wiederum Schaltflächen zum Öffnen weiterer Formulare zum Bearbeiten der Daten enthält.

Vielleicht hat Deine Anwendung auch ein Formular, das so oft verwendet wird, dass Du es standardmäßig direkt beim Öffnen anzeigen möchtest. Der einzige Nachteil dieser Methode ist, dass Du hier nur den Formularnamen angeben kannst, aber nicht die Optionen, die Du im Makrobefehl **ÖffnenFormular** oder in der Methode **DoCmd.OpenForm** findest.

Wenn Du diese Optionen beim Starten nutzen möchtest, kannst Du aber auch ein Makro definieren, dass beim Start ausgelöst wird und die Aktion **ÖffnenFormular** nutzt. Dieses Makro muss den Namen **AutoExec** tragen, damit es automatisch ausgeführt wird. Oder Du rufst in diesem Makro die Aktion **Ausführen Code** auf und übergibst den Namen einer VBA-Funktion, die wiederum die Methode **DoCmd.OpenForm**

ausführt und so das gewünschte Formular öffnet. Das sollen allerdings Themen weiterer Artikel sein – hier konzentrieren wir uns auf den **DoCmd.OpenForm**-Befehl.

Öffnen per Ribbon-Schaltfläche

Die **DoCmd.OpenForm**-Methode können wir beispielsweise auch nutzen, wenn wir unserer Anwendung ein Ribbon mit Schaltflächen hinzufügen. Den dadurch ausgelösten Prozeduren können wir ebenfalls die Methode **DoCmd.OpenForm** hinzufügen, um per Ribbon-Schaltfläche Formulare zu öffnen.

Makro oder VBA?

Die Makroaktion **ÖffnenFormular** und der VBA-Befehl **DoCmd.OpenForm** bieten grundsätzlich genau die gleichen Möglichkeiten. Die Parameter der Makroaktion sehen wir in Bild 1. Um diese Ansicht zu erhalten, legen wir mit **Erstellen|Makros und Code|Makro** ein neues Makro an und wählen als Aktion **ÖffnenFormular** aus.

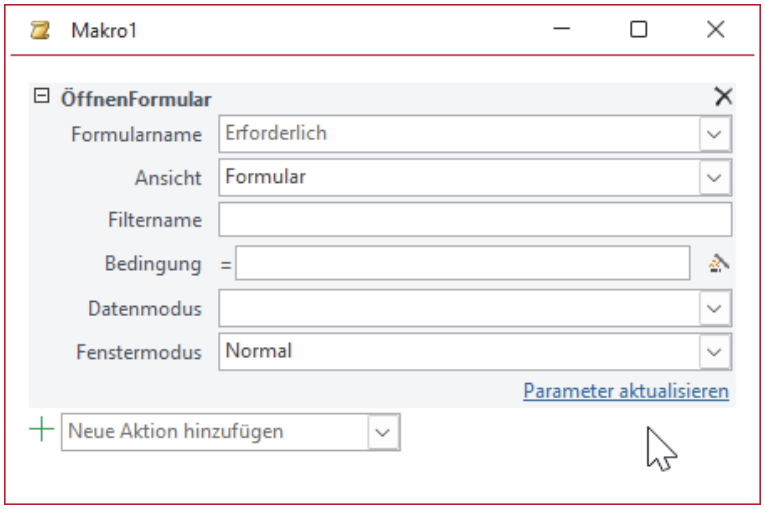

Bild 1: Die Makroaktion **ÖffnenFormular**

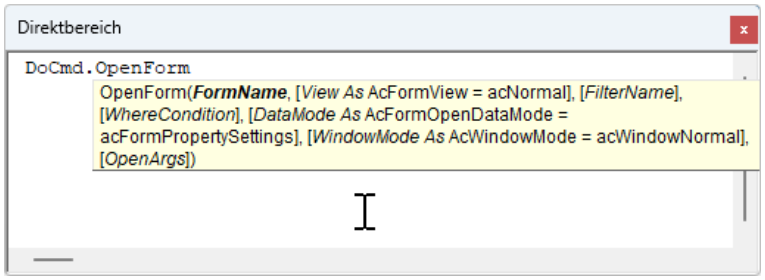

Bild 2: Der VBA-Befehl **DoCmd.OpenForm**

Hier sehen wir die Parameter, die wir in ähnlicher Form auch in der Parameterliste der VBA-Methode **DoCmd.OpenForm** vorfinden. Um diese einzusehen, öffnen wir den VBA-Editor und wechseln dort zum Direktbereich. Das gelingt vom Access-Hauptfenster am einfachsten mit der Tastenkombination **Strg + G**. Geben wir hier **DoCmd.OpenForm** gefolgt von einem Leerzeichen ein, erhalten wir die Parameterliste aus Bild 2. Schauen wir uns die Parameter der beiden Varianten in der Übersicht an:

- **Formularname** (**FormName**): Name des zu öffnenden Formulars
- **Ansicht** (**View**): Ansicht des Formulars beim Öffnen
- **Filtername** (**FilterName**): Name eines beim Öffnen anzuwendenden Filters
- **Bedingung** (**WhereCondition**): Angabe einer Bedingung in Form eines Kriteriums wie **AutorID = 1**
- **Datenmodus** (**DataMode**): Datenmodus, zum Beispiel neuer Datensatz, Datensatz bearbeiten, schreibgeschützt
- **Fenstermodus** (**WindowMode**): als modaler Dialog, ausgeblendet et cetera

Die VBA-Methode **DoCmd.OpenForm** liefert allerdings noch einen weiteren Parameter namens **OpenArgs**. Damit können wir beim Öffnen eine Zeichenkette übergeben, die wir in den Ereignissen des Formulars auswerten können.

Wir schauen uns diese Parameter in den folgenden Abschnitten für den Einsatz unter VBA an. VBA ist flexibler als Makros.

Testen der verschiedenen Parameter

Zum Ausprobieren der folgenden Anweisungen nutzen wir zunächst das Direktfenster des VBA-Editors. Hier können wir die Befehle einfach eingeben und ausprobieren. IntelliSense unterstützt uns bei der Angabe der Parameter und bei einigen Parametern werden auch die zur Verfügung stehenden Werte angezeigt.

In einem Punkt liefert die Verwendung der Makrofunktion **ÖffnenFormular** einen Vorteil gegenüber der VBA-Variante **DoCmd.OpenForm**: Der Makro-Editor bietet alle zur Verfügung stehenden Formulare zur Auswahl an (siehe Bild 3).

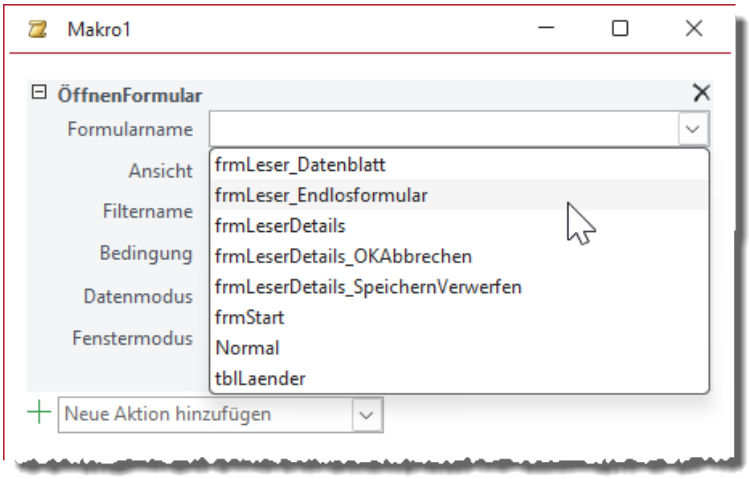

Bild 3: Die Makroaktion **ÖffnenFormular** bietet die Formulare zur Auswahl an.

Einfaches Öffnen eines Formulars

Die Mindestanforderung an die Methode **DoCmd.OpenForm** beim Öffnen eines Formulars ist die Angabe des Formularnamens. Wenn wir beispielsweise das Formular **frmLeserDetails** öffnen wollen, nutzen wir die folgende Anweisung:

```
DoCmd.OpenForm "frmLeserDetails"
```

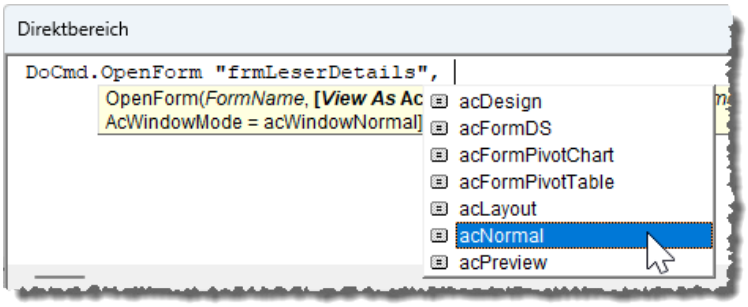

Bild 4: Werte für den Parameter **View**

Hier geben wir nur den Namen des zu öffnenden Formulars für den ersten Parameter **FormName** an. Diesen können wir einfach aus dem Navigationsbereich übernehmen, wir brauchen ihn nur noch in Anführungszeichen einzufassen. Die Angabe des Formularnamens ist Pflicht – woher sollte die Anweisung sonst wissen, welches Formular geöffnet werden soll? Die übrigen Parameter sind alle optional, teilweise werden Standardwerte verwendet.

Einstellen der Ansicht beim Öffnen eines Formulars

Mit dem zweiten Parameter **View** stellen wir ein, wie das Formular angezeigt werden soll. Es gibt die folgenden Werte, die per IntelliSense zur Auswahl angeboten werden (siehe Bild 4):

- **acDesign**: Öffnet das Formular in der Entwurfsansicht. Diese Option nutzt man bei Anwendungen, die man an den Benutzer weitergibt, nicht. Sie ist dafür gedacht, wenn man den Formularentwurf per Code ändern möchte – beispielsweise, um automatisiert Steuerelemente hinzuzufügen.

- **acFormDS**: Öffnet das Formular in der Datenblattansicht. Das ist die Ansicht, in der auch Tabellen standardmäßig erscheinen.

- **acFormPivotChart**, **acFormPivotTable**: Veraltete Parameterwerte, die unter aktuellen Access-Versionen nicht mehr funktionieren

- **acLayout**: Öffnet das Formular in der Layout-Ansicht. Diese Ansicht ist ein Hybrid zwischen der