Grundlagen des Datenbankdesigns

Wie entwerfe ich eine Datenbank? 4.1

Wenn Sie eine Datenbank erstellen wollen, beginnt das eigentliche Problem schon lange vorher: Wie kann die vielschichtige Wirklichkeit in ein starres Tabellenkonzept gepresst werden? Wenn Sie sich hier allzu schnell an den Computer setzen und schon mal ein paar Daten eintippen, kommt das böse Erwachen meist sehr bald. Datenbanken verzeihen nichts.

In Word können Sie gerne in völliger Ahnungslosigkeit alle Absätze rechtsbündig »formatieren«, indem Sie sie manuell links mit Leerzeichen auffüllen und jedes Zeilenende per ↵-Taste bestimmen. Das lässt sich jederzeit leicht mit ein oder zwei Makros korrigieren, ohne dass etwas verloren geht.

In Access – genauso wie in anderen relationalen Datenbanken – verlieren Sie Informationen, wenn das Datenbankdesign nicht stimmt. Daher müssen Sie schon in dieser Phase sehr sorgfältig arbeiten, auch wenn ich gut verstehen kann, dass ein bunter Formular-Assistent am Anfang vielleicht mehr Spaß macht als das Planen eines Konzepts.

Aber es zahlt sich aus, obwohl Sie immerhin etwa 20 bis 30 % des Gesamtaufwands für eine Datenbank in die Vorüberlegungen stecken sollten. Ihr Konzept sollten Sie unbedingt auf Papier (am besten eignet sich querformatiges DIN-A3 oder größer) erstellen und nicht schon mit Access.

Zu diesem Zeitpunkt verführt ein Computer zu unnötiger Exaktheit und engt Sie ein. Machen Sie also in der Skizze lieber einen Kringel und schreiben Sie erst einmal Kundendaten hinein, das reicht völlig. Architekten haben zu diesem Zeitpunkt einen 6B-Bleistift in der Hand. Der ist so dick, dass man sich auf das Wesentliche konzentrieren muss und sich gar nicht in Details verlieren kann.

Sollten Sie jetzt in Access bereits im Tabellenentwurf einzelne Felder benannt haben, würden Sie wegen des Arbeitsaufwands schon viel länger zögern, sie alle zu verschieben, zu löschen oder umzubenennen. Auf dem Papier hingegen reicht ein flotter Strich oder ein erläuternder Pfeil, wenn Sie Ihr Konzept umstrukturieren müssen. Nehmen Sie also ein Papier und machen ein erstes Brainstorming:

- Welche Informationen muss ich speichern? (Aber nicht: Wo muss ich sie speichern? Das gehört in die nächste Phase!)
- Bleiben die Dateninhalte immer unverändert? Und wenn nicht: Benötige ich die bisherigen Daten noch?

- Was soll auf dem Bildschirm angezeigt werden? Was muss ausgedruckt werden?
- Woher kommen die Daten? Muss ich Importdaten berücksichtigen?
- Was erfolgt durch manuelle Eingabe? Gibt es Schnittstellen zu anderen Programmen?
- Wie viele Benutzer arbeiten an der Datenbank? Wie viele davon gleichzeitig?
- Welche Datenmengen sind wo zu erwarten?
- Wird die Datenbank lokal, zentral oder gar synchronisiert mit Laptops benutzt?
- Wer macht was in der Datenbank? Haben alle die gleichen Rechte und müssen alle die gleichen Daten sehen?

Beim Niederschreiben werden sich sowohl weitere Fragen als auch gelegentliche Widersprüche ergeben. Das ist nicht schlimm, vor allem nicht in dieser Phase. Aber es ist ein schmerzlicher Unterschied, ob Sie jetzt nur ein Konzeptpapier zerreißen oder kurz vor der endgültigen Fertigstellung die gesamte Datenbank löschen.

Planungsphasen

Die Phasen zur Erstellung einer Datenbank werden im Englischen recht einprägsam als *Storming, Norming and Performing* benannt:

- **Storming:** ist das Brainstorming am Anfang, bei dem Sie ziemlich unstrukturiert alles an Ideen notieren, was Ihnen so einfällt. Das können konkrete Feldinhalte ebenso sein wie Sonderwünsche zu Formularen oder spezielle Design-Vorgaben für einen Bericht.
- **Norming:** fasst diese Ideen sinnvoll zusammen und sortiert dabei vor allem Felder zu ihren Tabellen
- **Performing:** schließlich ist die Verbesserung, wenn alles funktioniert. Dann können Sie sich darum kümmern, ob es mit Optimierungen noch ein wenig schneller geht.

Schreiben Sie zuerst auf, was Sie alles speichern müssen. Wenn es Ihnen hilft, können Sie dazu farbige Papierzettel nehmen, die in sogenannten Moderationskoffern angeboten werden. Einfache Notizzettel für je einen Feldnamen oder Beispieldaten tun es aber auch.

Fangen wir direkt einmal an: Sie möchten eine Adressenliste speichern. Welche Daten sollen darin gespeichert werden? Eine sinnvolle erste Auswahl könnte die in Abbildung 4.1 erstellte Liste sein.

Abbildung 4.1: Die vorläufige Liste der benötigten Informationen

In diesem Fall würde ich übrigens direkt noch eine *Priorität* als zu speichernde Information vorschlagen. Diese enthielte eine Angabe wie *aktuell*, *selten* oder *gelegentlich*. Anhand eines solchen Filters können Sie später beispielsweise nur aktuelle Adressen ausdrucken ohne den Ballast selten benötigter Adressen (etwa von alten Schulfreunden).

Daten inhaltlich trennen

Bevor Sie die Storming-Phase abschließen, sollten Sie noch einmal analysieren, wie viele Informationen im jeweiligen Feld stehen. Im *Name*-Feld steht beispielsweise *Maria Schmitz*, also der Vor- und Nachname.

Das aber ist mehr als lästig, wenn Sie irgendwann per Datenbank an *Frau Schmitz* schreiben wollen und daher den Nachnamen getrennt benötigen. Sie können schließlich nicht einfach voraussetzen, dass das zweite Wort im Feld der Nachname ist, wie das Beispiel *Johann Sebastian Bach* zeigt.

Da ist es besser, Vor- und Nachname direkt in zwei getrennten Feldern zu speichern und erst im Bedarfsfall zu verketten. Zur Not sehen Sie auch noch ein weiteres Feld für den zweiten Vornamen, einen Spitznamen oder die im Amerikanischen üblichen *junior/senior*-Zusätze vor.

Daten gemeinsam speichern

Für die Straße und Hausnummer gilt übrigens die entgegengesetzte Überlegung. Wenn Sie nicht gerade einen Routenplaner entwerfen, werden Sie Straße und Hausnummer immer gemeinsam benötigen. Natürlich könnten Sie auch diese zwei aus getrennten Feldern verketten. Aber sobald es etwa eine amerikanische Adresse ist, steht die Hausnummer vor der Straßenbezeichnung wie in *1600 Pennsylvania Avenue Northwest*, der Adresse des Weißen Hauses in Washington.

In einem einzigen Feld kann die Adresse dann direkt in der benötigten Reihenfolge notiert werden. Bei getrennten Feldern für Straße und Hausnummer müssten Sie hingegen beim Zusammensetzen eine Liste bestimmter Länder beachten.

Apropos Länder: Ist Ihnen aufgefallen, dass bisher gar kein Land zu speichern wäre? Manches fällt erst auf, wenn in Gedanken oder auf dem Papier mal ein paar Testdaten eingegeben werden. Es sollte also noch ein *Land*-Feld wie in Abbildung 4.2 ergänzt werden.

Tabelle 1
Vorname
Nachname
Land
PLZ
Ort
Straße
Telefon
Priorität

Abbildung 4.2: Die endgültige Liste der benötigten Informationen

Felder in Tabellen gruppieren

Nach dem *Storming* kommt das *Norming*, also die Entscheidung, welche Felder in welche Tabellen gehören. Hier wird sicherlich eine Tabelle *Personen* einzurichten sein. Dann müssen Sie nur noch für jeden zu speichernden Datenwert entscheiden, ob er zu einem Feld dieser Tabelle wird. Das können Sie dann bejahen, wenn der Datenwert eine Eigenschaft des Tabellenobjekts ist.

Beim Vornamen ist das sicherlich uneingeschränkt zu bestätigen, denn der Vorname kann als untrennbare Eigenschaft einer Person bezeichnet werden. Auch den Nachnamen könnte man als direkt mit der Person verbunden sehen. Aber so fest nun auch wieder nicht, denn der Nachname kann sich ja durch Heirat oder Scheidung ändern.

Es ist nun eher eine *politische* als eine datenbanktechnische Entscheidung, ob Sie den Nachnamen als Eigenschaft der Person betrachten. Solange Sie nur der jeweils aktuelle Nachname interessiert, können Sie das Tabellendesign nämlich so belassen.

Erst wenn diese Tabelle wie beispielsweise für das Einwohnermeldeamt auch vorherige Nachnamen speichern muss, dürfte dieses Feld nicht direkt in der *Personen*-Tabelle gespeichert werden. Mit diesem Historienproblem werden wir uns noch ausführlicher befassen. Daher bleibt es erst einmal dabei, dass alle genannten Felder in einer einzigen Tabelle enthalten sind.

Zweite Normalform

Leider haben Sie damit schon eine der Grundregeln für die Erstellung relationaler Datenbanken verletzt. Damit keine Werte mehrfach vorhanden sind oder sich widersprechen können, gibt es Regeln, die als *Normalformen* einer Datenbank bezeichnet werden. Sie wurden in den 1960er Jahren von Frank Edgar Codd, einem Mathematiker am IBM Almaden Research Center in Kalifornien, entwickelt.

Die gleichzeitige Speicherung von PLZ und Ort in einer Tabelle bedeutet, dass der Benutzer später nach Belieben voneinander unabhängige Daten eintragen darf. Stellen Sie sich vor, die Daten würden wie in Tabelle 4.1 ausgefüllt.

Abgesehen davon, dass es sich sowieso um fiktive Adressen handelt, ist in Aachen die Postleitzahl 52077 durchaus möglich. In München jedoch beginnen alle Postleitzahlen mit *80…*, daher kann *52088* nicht stimmen.

Feld	Adresse 1	Adresse 2
Vorname	Maria	Hans
Nachname	Schmitz	Meier
Land	D	D
PLZ	52077	52088
Ort	Aachen	München

Feld	Adresse 1	Adresse 2
Straße	Hauptstraße 1	Stachus 99
Telefon	0241/998877	089/123456
Priorität	aktuell	alt

Tabelle 4.1: Die Beispieldaten für die Personentabelle

Die zweite (ja, die erste wird noch nachgereicht!) Normalform besagt: Daten, die voneinander abhängig sind, dürfen nicht in der gleichen Tabelle stehen. Natürlich ist das im Original sehr mathematisch und vielfach komplizierter formuliert, aber für unsere Zwecke reicht es so. Das Stichwort heißt an dieser Stelle *Redundanz.*

Haben Sie die Verletzung gesehen? Der Ort ist in Wirklichkeit von der PLZ abhängig! Sobald Sie die PLZ hinschreiben, ist der Ort eindeutig vorgegeben, daher dürfen Sie diesen nicht mehr frei eintragen. Vielmehr müssten Sie ihn aus einer sogenannten Nachschlagetabelle heraussuchen.

Entscheidung zwischen Theorie und Praxis

Nach der Theorie kommen wir nun zum praktischen Teil. Denn leider müssen Sie sich häufig zwischen einer datenbanktheoretisch korrekt normalisierten Tabelle und einer guten Performance entscheiden.

In diesem Fall etwa bräuchte es eine mehrere MegaByte große Hilfstabelle aller deutschen Postleitzahlen und der zugehörigen Ortsbezeichnungen. Dem stehen geschätzte 100 bis 200 Adressen entgegen, die hier irgendwann einmal eingetragen und dann kaum 400 KiloByte groß sind. Außerdem sind auch andere Länder möglich, deren weltweite Postleitzahlen also ebenfalls nachgeschlagen werden müssten. Das würde die vermutliche Größe der PLZ-Nachschlagetabellen explosionsartig ansteigen lassen.

Daher gibt es auch hier wieder eine eher *politische* und weniger datenbanktechnische Betrachtung: Was würde passieren, wenn eine solche falsche Münchner Adresse eingetragen wird? Angenommen, mit dieser Adresse würde eine Rechnung versandt, könnte sie vom Postboten mit dem Vermerk *Adresse nicht gefunden* zurückgeschickt werden.

Dann wäre eigentlich alles in Ordnung, denn das ist ein Hinweis darauf, dass dieser Datensatz überprüft werden muss. Abgesehen davon, dass Sie dadurch immer noch keine richtige Adresse kennen, gibt es damit einen Mechanismus, der diesen Fehler kennzeichnet.

Wenn der Postbote jedoch sehr engagiert ist, stellt er vielleicht die ungültige PLZ/Ort-Kombination fest, streicht die falsche PLZ und findet die richtige heraus. Dann wird der Brief neu verteilt und landet in München beim gemeinten Empfänger. Auch dann wäre alles in Ordnung, die Rechnung ist ja trotz des Fehlers angekommen.

Angesichts des geringen Risikos und der gigantisch großen PLZ-Nachschlagetabellen im Vergleich zu den Adressen werde ich an dieser Stelle ausdrücklich den Bruch mit der zweiten Normalform hinnehmen. PLZ und Ort sind redundant, aber das ist hier sinnvoll.

Erste Normalform

Damit komme ich auf die erste Normalform zurück. Diese besagt sinngemäß: Mehrere Dateninhalte in einem Feld sind nicht zulässig. Das Stichwort hier heißt *Atomisierung*.

Natürlich haben Sie recht, dass in den Beispieldatensätzen keine mehrfachen Daten in einem Feld enthalten sind. Aber was machen Sie, wenn jemand mehrere Telefonnummern hat? Da das durchaus nicht ungewöhnlich ist, müssen Sie Handy, Festnetz, Fax irgendwo speichern können.

Es hilft nichts, wenn Sie schnell die Felder *Handy* und *Fax* hinzunehmen. Schließlich haben viele auch mehrere Handy- oder Faxnummern. Bitte machen Sie jetzt auch nicht den weit verbreiteten Fehler, einfach ein paar Felder *Handy1*, *Handy2*, *Handy3*, *Fax1* und *Fax2* bereitzustellen. Das ist garantiert immer ein Feld zu wenig.

Auch dieses Feld müsste in eine separate Tabelle mit Abhängigkeit von dieser Personentabelle. In einer solchen Telefontabelle kann dann pro Zeile eine Telefonnummer angegeben werden, also beliebig viele je Person.

Nur zur Vereinfachung wird unsere erste Tabelle nur eine Telefonnummer je Person speichern können, die Atomisierung von Daten wird selbstverständlich später an einem anderen Beispiel durchgeführt.

Tabellen- und Feldnamen

Sobald es konkret wird und die Tabellen/Felder festgelegt sind, sollten Sie sich auch Gedanken über die wirklichen Namen machen. Access lässt Ihnen da sehr viele Freiheiten, denn Namen dürfen bis zu 64 Zeichen lang sein.

Namen dürfen allerdings keine eckigen Klammern ([]), kein Ausrufezeichen und keinen Punkt enthalten. Ausrufezeichen und Punkt sind reserviert als Trennzeichen zwischen Objekt und Unterobjekt wie in `Me.Controls.Count` oder `Tabelle1!Name`. Die Klammern werden benötigt, wenn die Namen Leerzeichen enthalten, um den kompletten Namen einzufassen, beispielsweise als `[Stammdaten Kunden]!ID`.

Daraus folgt schon, dass alle Zeichen außer *[].!* erlaubt sind, wie bereits gesehen auch das Leerzeichen. Trotzdem würde ich von vier weiteren Zeichen abraten:

- Vermeiden Sie Leerzeichen in Namen. Sie müssten sonst solche Namen immer mit eckigen Klammern einschließen, was den Arbeitsaufwand unnötig erhöhen würde.
- Vermeiden Sie das Minuszeichen (-). Das, was die meisten für einen Bindestrich (Divis, halten, ist in Wirklichkeit ein Minuszeichen. In Abfrageformeln wird etwa ein Feld namens Kauf-Datum automatisch mit Feldklammern versehen: `[Kauf]-[Datum]`. Dann sucht die Abfrage anschließend nach den beiden Feldern Kauf und Datum, um sie voneinander abzuziehen. Gemeint war aber `[Kauf-Datum]`, was Sie bei jeder neuen Abfrage manuell korrigieren müssten.

- Das Gleiche gilt für den Schrägstrich (/), der ebenfalls missverständlich ist. So wird aus dem Feld Ein/Ausgabe in Abfragen automatisch `[Ein]/[Ausgabe]` und damit der aussichtslose Versuch, das Feld Ein durch das Feld Ausgabe zu teilen.

Der Unterstrich (_) hingegen ist ein normales Zeichen, das manchmal zur optischen Unterteilung langer Worte hilfreich ist. In diesem Buch wird die sogenannte Kamelschreibweise benutzt, bei der Großbuchstaben mitten im Wort die Lesbarkeit verbessern. Das ist kürzer als die Konstruktion mit dem Unterstrich.

Wichtig: Namenskonventionen

Namen dürfen bis zu 64 Zeichen lang sein, aber keine eckigen Klammern, Ausrufezeichen oder Punkte enthalten. Das gilt sowohl für Feldnamen als auch für alle anderen Namen von Tabellen, Abfragen, Formularen, Berichten, Makros und Modulen.

Außerdem ist zu empfehlen, auf andere Sonderzeichen wie Leerzeichen, Minuszeichen oder Schrägstrich möglichst zu verzichten, weil deren Sonderbehandlung mindestens lästig ist. Vor allem handeln Sie sich damit unnötige Fehlerquellen ein.

Namenskonventionen

Die hier benutzten Namen für Datenbankobjekte basieren auf der von Stan Leszynski und Gregory Reddick entwickelten Namenskonvention, die wiederum die Ideen der Ungarischen Notation für den Einsatz in heutigen Programmen aufgegriffen hat. Die Ungarische Notation des ungarischen ehemaligen Microsoft-Programmierers Charles Simonyi ist ursprünglich als eine Methode gedacht gewesen, Variablen möglichst eindeutig und vor allem einheitlich zu benennen.

Beiden gemeinsam ist die Tatsache, dass jedem Namen der Typ des benannten Objekts vorangestellt wird; bei Variablen zusätzlich noch die öffentliche oder private Sichtbarkeit. Der Typ wird dabei als meist dreibuchstabige Abkürzung der englischen Bezeichnung gewählt. Tabellennamen beginnen also mit *tbl* für *table*, Abfragen mit *qry* für *query*.

Tabelle 4.2 zeigt den jeweiligen Namensanfang (Präfix) für die in Access vorkommenden Objekte.

Präfix	Objekt
tbl	Tabelle (engl.: table)
qry	Abfrage (engl.: query)
frm	Formular (engl.: form)
rpt	Bericht (engl.: report)
mac	Makro (engl.: macro)
mod	Modul (engl.: module)

Tabelle 4.2: Mögliche Präfixe für Access-Objekte

Charles Simonyi

Wenn Sie das Gefühl haben, dass Sie den Namen Charles Simonyi schon mal gehört haben: Ja, er ist derjenige, der im April 2007 und im März 2009 als bisher einziger Weltraumtourist bereits zwei Mal zur Internationalen Raumstation ISS mitgeflogen ist. Zwanzig Jahre vorher war er etwas weniger spektakulär als Chefentwickler bei Microsoft für die Entwicklung von Word und Excel zuständig.

Damit wird der Name unserer vorhin entwickelten Kundentabelle *tblPersonen* sein. Hier empfehle ich übrigens eine weitere Vereinheitlichung, die nicht als Access-Vorgabe zwingend notwendig, aber sehr hilfreich ist: Tabellennamen nennen bei mir das beschriebene Objekt immer im Plural, weil darin viele Daten enthalten sind. Es heißt also dann nicht *tblBestellung*, sondern *tblBestellungen*.

Feldnamen hingegen wähle ich immer im Singular, weil darin jeweils nur ein Datenwert enthalten ist. Das gilt sogar für die mehrwertigen Felder, bei denen im Feld scheinbar mehrere Inhalte gespeichert werden (in Wirklichkeit handelt es sich nur um eine verkappte m:n-Beziehung.).

Letzten Endes ist es egal, an welches Schema Sie sich halten, Sie können selbstverständlich auch alle Objekte im Singular benennen. Hauptsache, es ist einheitlich, denn Sie werden gelegentlich die Namen aus dem Gedächtnis heraus (oder von Ihrem Konzeptpapier!) wissen müssen. Da ist es am einfachsten, sich nicht auch noch mit solchen Kleinigkeiten herumzuschlagen.

Die Feldnamen führen jedoch ausnahmsweise als Präfix nicht ihren Typ, sondern eine Abkürzung des Tabellennamens. Damit sind sie datenbankweit eindeutig. Das ist zwar in Access ebenfalls nicht technisch notwendig, aber bedeutend übersichtlicher.

Alle Felder in *tblPersonen* beginnen also mit *per*, sodass es in dieser gesamten Datenbank kein gleichnamiges Feld mehr geben kann. Das lohnt sich, sobald in mehreren Tabellen ein Namensfeld benötigt wird. Wenn Sie dann nämlich mehrere Tabellen in Abfragen miteinander verknüpfen, heißt es bei gleichnamigen Feldern:

```
… WHERE tblArtikel.Name = "xyz" OR tblPersonen.Name = "abc"
```

Eindeutige Felder ermöglichen das Weglassen der Tabellenbezeichnungen:

```
… WHERE artName = "xyz" OR perName = "abc"
```

Auch in den angezeigten Überschriften von Tabellen- oder Abfragespalten lässt sich ein Feld mit dem Namenspräfix eindeutiger zuordnen.

Die bisher festgelegten Felder werden also wie in Abbildung 4.3 benannt, wobei ich aufgrund schlechter Erfahrungen mit früheren Access-Versionen und dem Einsatz von Umlauten oder *ß* auch auf diese Zeichen in Namen grundsätzlich verzichte.

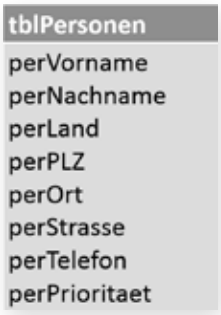


Abbildung 4.3: Die endgültigen Feldnamen für die Personentabelle

Tabelle anlegen

Nach diesen Vorüberlegungen ist es endlich an der Zeit, die Tabelle anzulegen. Das soll natürlich nicht in der Nordwind-Datenbank geschehen, die möglicherweise noch geöffnet ist.

1. Die zentralen Tätigkeiten wie Speichern, Schließen und Öffnen sind bei Access 2013 in der sogenannten Backstage-Ansicht untergebracht. Klicken Sie daher bitte in der linken oberen Ecke auf die Registerkarte *DATEI*. Dort erscheint die Backstage-Ansicht wie in Abbildung 4.4, von der aus Sie sowohl die aktuelle Datenbank schließen als auch eine neue anlegen können.

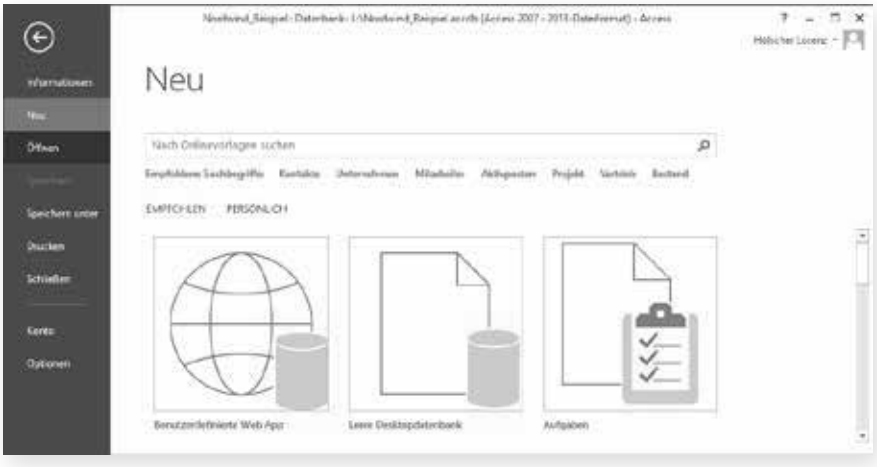


Abbildung 4.4: Die Backstage-Ansicht erreichen Sie über die Registerkarte DATEI

2. Wählen Sie dort *Neu* und im mittleren Teil des Bildschirms *Leere Desktopdatenbank*.

3. Geben Sie dann als Dateinamen im erscheinenden Eingabefeld Adressen ein und wählen Sie mit einem Klick auf das daneben sichtbare Verzeichnissymbol ein geeignetes Verzeichnis aus. Mit anschließendem Klick auf die Schaltfläche ERSTELLEN wird die Datenbank angelegt und direkt eine neue Tabelle in der Datenblattansicht geöffnet.

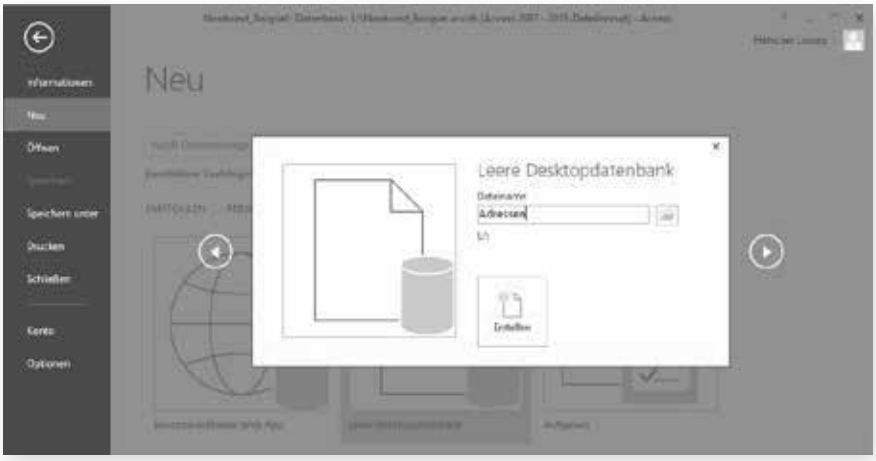


Abbildung 4.5: Geben Sie hier den gewünschten Dateinamen ein

4. Da wir die Tabelle viel detaillierter vorgeben wollen als die Datenblattansicht das könnte, wechseln Sie bitte wie in Abbildung 4.6 gezeigt per Rechtsklick auf die Registerkarte *Tabelle1* und die Auswahl des Befehls *Entwurfsansicht* in die Entwurfsansicht.

Abbildung 4.6: Wechsel in die Entwurfsansicht

5. Da Sie nun die Tabelle direkt speichern müssen, geben Sie ihr wie besprochen bitte den Namen tblPersonen. Darin finden Sie noch das von Access automatisch in der Datenblattansicht vorgegebene Feld ID, das Sie auf dem Zeilenkopf (mit dem Schlüsselsymbol) anklicken und mit der Entf-Taste löschen können.

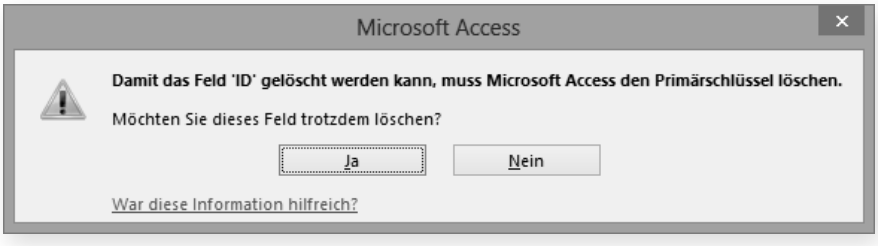


Abbildung 4.7: Bestätigen Sie das Löschen des Primärschlüssels

6. Die Rückfrage, dass dazu auch der Primärschlüssel gelöscht werden muss, können Sie mit Ja bestätigen, denn wir werden gleich sowieso einen neuen einfügen. Jetzt ist der Tabellenentwurf leer und Sie können die Felder eingeben, wie sie auf Seite 59 in Abbildung 4.3 aufgelistet sind.

Felddatentypen

Es beginnt mit dem ersten Feld *perVorname*, das erwartungsgemäß den Felddatentyp *Kurzer Text* erhält. Die Anzahl der zulässigen Zeichen kann unten in den Feldeigenschaften als *Feldgröße* angegeben werden, maximal zulässig sind *255*. Da dies für die meisten Vornamen viel zu lang sein dürfte, schlage ich hier *30* als *Feldgröße* vor.

So markieren Sie schneller

Um solche Werte in diesem und anderen Eigenschaftenfenstern zu ändern, klicken Sie am besten genau nicht in das eigentliche Eingabefeld, sondern in die links davon stehende Überschrift. Dann wird der komplette Wert markiert und Sie können ihn überschreiben. Das ist natürlich in diesem Fall ein unnötiger Aufwand, weil sich die kurze Zahl auch leicht direkt mit der Maus markieren lässt.

Es gibt aber viele Eigenschaften, wo ein solcher Wert sehr lang ist und damit nicht komplett angezeigt werden kann. Das ist beispielsweise bei SQL-Anweisungen der Fall. Dann ist trotzdem durch Klick auf die Beschriftung direkt alles markiert.

Eine solche Feldgröße beim Felddatentyp *Kurzer Text* bedeutet, dass der Benutzer später 30 Zeichen eingeben kann. Beim 31. Zeichen reagiert Access dann mit einem Piepsen. Es werden also erst gar keine überzähligen Zeichen angenommen, die beim Speichern verworfen werden müssten.

Sie können ein solches Feld jederzeit vergrößern und mehr Zeichen zulassen, falls Sie oder die Benutzer feststellen, dass es zu knapp bemessen war. Selbstverständlich können Sie es auch verkleinern. Aber dann wird Access Sie bloß eher allgemein warnen, dass dabei möglicherweise Daten gelöscht werden, ohne detailliert zu sagen, ob und welche Datensätze betroffen sind. Sie müssten dann zuerst (mit später noch zu zeigenden Abfragen) die betroffenen Datensätze herausfiltern und prüfen, welche Buchstaben denn tatsächlich gelöscht würden. Das ist einfach mehr Aufwand. Deshalb empfehle ich, die Zeichenanzahl bei Texten durchaus knapp zu kalkulieren, vor allem, weil die Text-Datentypen gemessen an Zahlen-Datentypen sehr viel Platz benötigen.

Abbildung 4.8: ***Die Eigenschaften des ersten Felds*** perVorname

Abbildung 4.8 zeigt die Tabelle mit dem ersten Feld. Auch *perNachname* können Sie nach dem gleichen Muster eintragen, wegen der häufigen Doppelnamen würde ich hier aber eher *50* als *Feldgröße* nehmen.

Text-Feldtypen

Die bisherigen Datentypbezeichnungen *Text* und *Memo* sind seit dieser Version in *Kurzer Text* und *Langer Text* umbenannt worden.

Das Feld *perLand* mit ebenfalls dem Felddatentyp *Kurzer Text* sollte mit drei Zeichen ausreichend dimensioniert sein, da hier nur die postalischen Abkürzungen wie *D*, *NL* oder *CH* gespeichert werden.

Geben Sie dort für die Eigenschaft *Standardwert* am besten `D` ein, weil die meisten Adressen voraussichtlich aus Deutschland sind. Access wird diese uneindeutige Angabe direkt als „`D`" korrigieren, denn es ist tatsächlich die Zeichenkette *D* gemeint. Tabelle 4.3 zeigt die verschiedenen Möglichkeiten, in Standardwerten oder auch in SQL-Befehlen die gemeinten Datentypen zu kennzeichnen.

Eingabe	Bedeutung
„Datum"	Eine Zeichenkette mit den Buchstaben „Datum"
=Datum()	Eine Funktion namens *Datum* ohne Parameter, daher sind die runden Klammern dahinter leer. Die Klammern sind trotzdem notwendig.
[Datum]	Ein Feld namens *Datum*, dessen aktueller Inhalt an dieser Stelle benutzt wird
#24.12.2013#	Ein Datum (nämlich Weihnachten 2013) in der ländertypischen Reihenfolge, wie sie in der Windows-Systemsteuerung festgelegt wird

Tabelle 4.3: Die möglichen Eingabekennzeichnungen

Da Sie während des Entwurfs – und wir befinden uns ja gerade in einem Tabellenentwurf – für das Speichern zuständig sind, sollten Sie zwischendurch immer mal Strg + S drücken oder das Diskettensymbol ganz oben links anklicken.

Das nächste Feld wäre *perPLZ*, also die Postleitzahl. Da es sich hier vermutlich nicht um ein Textfeld, sondern um viel platzsparender zu speichernde Zahlen handelt, möchte ich Ihnen in Tabelle 4.4 kurz die möglichen (Zahlen-)Datentypen zeigen.

Felddatentypen	Zahlenbereich	Speicherbedarf	Bemerkungen
Ganzzahlige Datentypen			
Ja/Nein (Bit)	0 -1	1 Byte = 8 Bit	Der Ausdruck wird entweder als Wahr (= -1) oder Falsch (= 0) gewertet. Die erstaunliche Größe (8 Bit statt eigentlich 1 Bit) liegt an der Kompatibilität zu anderen Datenbanken und dem möglichen Null-Zustand.

Felddatentypen	Zahlenbereich	Speicherbedarf	Bemerkungen
Byte	0-255	1 Byte = 8 Bit	Datentyp für kleine positive Ganzzahlen
Integer	-32.768 .. +32.767	2 Byte = 16 Bit	Datentyp für mittelgroße Ganzzahlen
Long (Integer)	-2,1 Milliarden .. +2,1 Milliarden	4 Bytes = 32 Bit	Datentyp für große Ganzzahlen, der standardmäßig auch für AutoWerte benutzt wird
Währung	-922 Billionen .. + 922 Billionen	8 Bytes = 64 Bit	Eigentlich intern eine Ganzzahl, denn der Wert wird mit 10.000 multipliziert und gespeichert, was faktisch jedoch eine Festkommazahl mit 15 Vorkomma- und 4 Nachkommastellen ergibt
Nachkomma-Datentypen			
Datum/Zeit	01.01.100 .. 31.12.9999	8 Bytes = 64 Bit	Speicherbar sind nur gültige Datums- oder Zeitwerte. Der Wert 1 entspricht (wegen eines Schaltjahr-Berechnungsfehlers) Silvester 1899, die 2 dem Neujahrstag 1900, usw. Nach dem Komma werden Prozente von 24 Stunden angegeben, also 0,5 = 12 Uhr mittags und 0,33 = 7:55:12 Uhr.
Single	Etwa -10^{38} .. $+10^{38}$	4 Bytes = 32 Bit	Gleitkommazahl mit einfacher Genauigkeit
Double	Etwa -10^{308} .. $+10^{308}$	8 Bytes = 64 Bit	Gleitkommazahl mit doppelter Genauigkeit
Dezimal	Etwa -10^{29} .. $+10^{29}$	12 Bytes = 96 Bit	Variable Gleitkommazahl mit wechselnder Genauigkeit
Sonstige Datentypen			
Kurzer Text	maximal 255 Zeichen	1 Byte (ANSII) 2 Bytes (Unicode)	Datentyp für Texte, sowohl im ANSII-Format als auch in Unicode
Langer Text	maximal 64.000 Zeichen		Wie Kurzer Text
Replikations-ID		16 Bytes = 128 Bit	Weltweit eindeutige Zeichenkombination
Anlage		variabel	Kann mehrere Dateien enthalten
Berechnet		variabel	Formel, um kompatibel zu SharePoint-Tabellen zu sein

*Tabelle 4.4: **Die Felddatentypen und ihre Größen***

Die Felder sind in der Reihenfolge ihres Platzbedarfs angegeben. Um eine Postleitzahl möglichst sparsam zu speichern, sollte also der kleinste Zahlen-Datentyp gewählt werden.

Leider erliegen hier viele der Versuchung, nur die Anzahl der Ziffern zu zählen und sich für *Integer* zu entscheiden. Das klappt nur nördlich von Bielefeld, weil dort die Postleitzahlen tatsächlich kleiner als 32.767 sind. In Sennestadt, einem Stadtteil von Bielefeld, hat die dortige PLZ 33689 aber schon die Obergrenze für Integer-Zahlen überschritten.

Wenn es also wirklich nur um deutsche PLZ-Werte ginge, die bis maximal 100.000 gehen könnten, wäre der Felddatentyp *Long* (den Zusatz *Integer* lässt man üblicherweise weg) der kleinstmögliche.

Tatsächlich aber gibt das Feld *perLand* bereits einen Hinweis darauf, dass auch nichtdeutsche Postleitzahlen vorkommen werden. Diese heißen beispielsweise in anglo-amerikanischen Ländern ZIP-Code, weil sie dort gar nicht nur aus Ziffern bestehen. Für den London Tower etwa lautet der ZIP-Code EC3N 4AB, enthält also nicht nur ein Leerzeichen, sondern vor allem auch Buchstaben. Entsprechendes gilt für andere britische Orte und für Kanada.

Damit hat sich jede Überlegung bezüglich eines Zahlen-Datentyps erledigt, es muss für *perPLZ* der Felddatentyp *Kurzer Text* sein, als Textlänge sind mindestens 7 Zeichen notwendig. Trotzdem war der Exkurs nicht umsonst, denn wir werden auf die Datentypen wieder zurückkommen.

Alle übrigen Felder können Sie nun ergänzen, sodass sich der Tabellenentwurf wie in Abbildung 4.9 präsentiert.

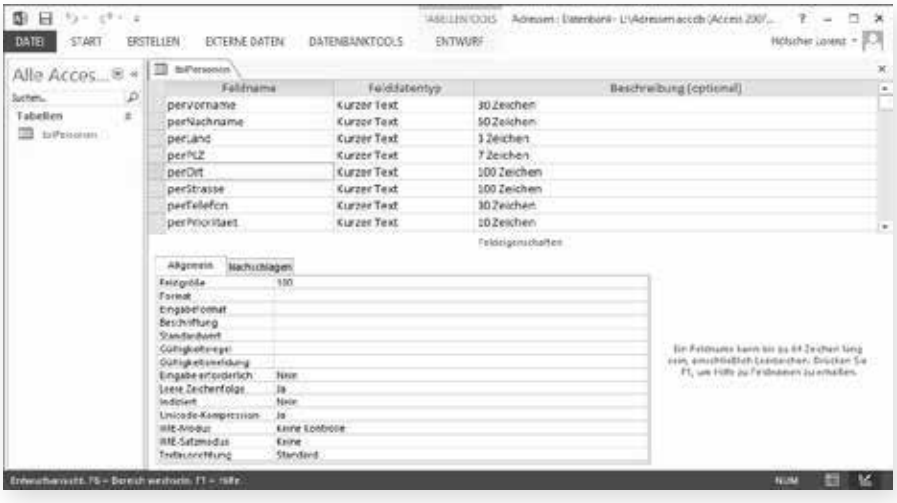


*Abbildung 4.9: **Die Felder im Entwurf der Personentabelle***

Primärschlüssel

Sie erinnern sich vielleicht noch, dass beim Löschen des Felds *ID* eine Rückfrage zum sogenannten *Primärschlüssel* kam. Der Primärschlüssel ist für jede Tabelle das Feld (oder die Felder, denn es können auch mehrere sein), das jeden beliebigen Datensatz eindeutig identi-

fiziert. Daher ist es mehr als sinnvoll, eigentlich fast zwingend, wieder einen Primärschlüssel anzugeben.

Welches Feld oder welche Felder wären bei Adressen dazu geeignet, in jedem Datensatz eindeutig zu sein? Der Vorname alleine ebenso wenig wie nur der Nachname. Selbst beide zusammen sind bei häufigen Namen wie *Michael Müller* oder *Gaby Meier* nicht ausreichend, in großen Städten wie etwa Berlin nicht einmal in Kombination mit einer Ortsbezeichnung.

Wenn Sie Pech haben, ist erst die gleichzeitige Berücksichtigung aller Feldinhalte eindeutig, wenn nämlich zwei gleichnamige Personen im gleichen Haus wohnen und nur unterschiedliche Telefonnummern haben. Das ist natürlich kaum praktikabel, sieben Felder gleichzeitig zu indizieren.

Also vergeben Sie einfach für jeden Datensatz eine eindeutige Nummer in einem zusätzlichen Feld. Genau genommen vergeben nicht Sie diese Nummer manuell, was durchaus möglich wäre, sondern Access macht das automatisch.

Die passende Einstellung *AutoWert* als *Felddatentyp* ist eigentlich nicht korrekt bezeichnet, denn der tatsächliche Datentyp ist ein *Long*-Wert, der standardmäßig bei 1 beginnt und für jeden neuen Datensatz um 1 erhöht wird. Der *AutoWert* ist also eine *Long*-Zahl mit automatischer Erhöhung.

Wenn Sie noch einmal in Tabelle 4.4 auf Seite 63 sehen, werden Sie feststellen, dass der Zahlenbereich für *Long* eigentlich bei -2,1 Milliarden beginnt. AutoWerte beginnen trotzdem immer bei 1 und haben daher schon die Hälfte der Möglichkeiten verschenkt. Es bleiben also nur noch 2,1 Milliarden mögliche Datensätze beziehungsweise Adressen übrig, das wäre ein Drittel der Weltbevölkerung. Das reicht, sagen Sie? In diesem Fall sicherlich.

Es ist aber trotzdem eine Überlegung wert, ob das genug ist, denn AutoWerte werden Sie regelmäßig einsetzen. Vielleicht brauchen Sie später mal Tabellen, bei denen zu Abrechnungszwecken Handy-Gesprächseinheiten mitprotokolliert werden. Wenn Sie Pech haben, sind die 2,1 Milliarden Einheiten da in einer halben Woche voll! Für diesen Fall könnten Sie deren *Feldgröße* noch auf *Replikations-ID* umstellen, die mit $3{,}4 \times 10^{38}$ Möglichkeiten ausreichend Reserve bieten wird.

1. Um ein neues Feld vor den bereits vorhandenen einzufügen, können Sie den Zeilenkopf der ersten Zeile, also von perVorname, anklicken und die Einfg-Taste drücken.
2. Dann geben Sie in der entstandenen Leerzeile perID als neues Feld ein und wählen AutoWert als Felddatentyp.
3. Wenn Sie nun in der Registerkarte *ENTWURF* das Primärschlüssel-Symbol anklicken, erscheint im Zeilenkopf wieder der kleine Schlüssel. Dadurch ändert sich unten in den Feldeigenschaften der Wert der Eigenschaft Indiziert von Nein auf Ja (Ohne Duplikate). Access wird dadurch selbstständig verhindern, dass jemals doppelte Werte auftreten.
4. Jetzt sollte die Tabelle wie in Abbildung 4.10 vollständig sein und kann mit Daten gefüllt werden.

Abbildung 4.10: Die endgültigen Feldnamen und -größen für die Tabelle tblPersonen *(KT30 = Kurzer Text mit 30 Zeichen)*

Daten eingeben

Damit Sie Daten in die neue Tabelle eingeben können, müssen Sie von der Entwurfsansicht wieder in die Datenblattansicht wechseln. Dazu gibt es, wie in Abbildung 4.11 zu sehen ist, drei Möglichkeiten:

- per Rechtsklick auf die Registerkarte *tblPersonen* und den Menübefehl *Datenblattansicht* oder
- durch Anklicken des Ansicht-Symbols ganz links in der *ENTWURF*-Registerkarte, das sich dabei jeweils automatisch umstellt und direkt als Umschalter zwischen den Ansichten nutzbar ist, oder
- mit einem Klick auf eines der Ansichtsymbole ganz unten rechts in der Statuszeile

Abbildung 4.11: Umschalten von der Entwurfsansicht zur Datenblattansicht

Um für die Dateneingabe von einer Zelle zur nächsten zu gelangen, können Sie sowohl die ↵ -Taste als auch die ↹ -Taste benutzen.

Wenn Sie nun in der Datenblattansicht Beispieldaten für die Adressen eingeben, werden Sie feststellen, dass sich das erste Feld *perID* zwar anklicken, aber nicht mit Werten füllen lässt. Das ist korrekt, denn es handelt sich dabei ja um das AutoWert-Feld, das seine Daten selbst erzeugt. Dabei fügt bereits der Beginn eines neuen Datensatzes die nächste AutoWert-Nummer ein, wie Sie in Abbildung 4.12 sehen können.

Es ist auch egal, ob Sie diesen Datensatz für *Volker Versuch* gar nicht mit Speichern beenden, indem Sie zweimal die Esc -Taste drücken. Dann wird er zwar nicht gespeichert, aber der nächste Datensatz erhält nur die nächsthöhere Nummer 6. Die Nummer 5 ist verfallen.

Das hat etwas damit zu tun, dass Access immer mit zulässigen Mehrbenutzerzugriffen auf diese Tabelle rechnet. Wenn also – während Sie den Datensatz mit *Volker Versuch* gerade

bearbeiten – jemand anderes auch einen neuen Datensatz beginnt, muss Access trotzdem eindeutige *perID*-Werte sicherstellen.

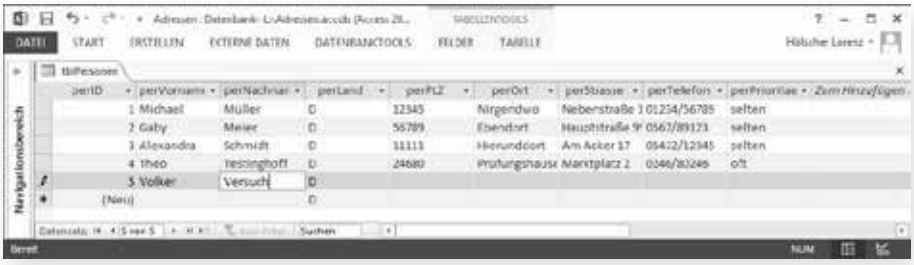


*Abbildung 4.12: **Erfassen des fünften Beispieldatensatzes***

Vielleicht ist Ihnen bei der Eingabe der Daten noch etwas aufgefallen? Es gab keine Aufforderungen zum Speichern! Anders als Sie es vielleicht von anderen Programmen gewohnt sind, kümmert sich Access um das Speichern jedes Datensatzes, sobald Sie ihn verlassen haben.

Das geschieht sowohl dann, wenn Sie per ↵-Taste vom letzten Feld *perPrioritaet* zum ersten Feld *perID* des folgenden Datensatzes wechseln, als auch beim Schließen des Tabellenfensters mit dem *X* am rechten Rand. Sie können also bei einem Computerabsturz maximal den einen Datensatz verlieren, den Sie in diesem Fenster noch nicht gespeichert haben, alle anderen wurden schon gesichert.

Einen noch ungespeicherten Datensatz erkennen Sie daran, dass er wie in Abbildung 4.12 links im Zeilenkopf noch den Schreibstift anzeigt. Bereits gesicherte Datensätze zeigen dort gar kein Symbol bzw. ein Dreieck für die jeweils markierte Zeile.

Die letzte Zeile mit dem Sternchen ist immer reserviert für die Eingabe neuer Datensätze. Hier werden schon die Standardwerte angezeigt, wie für *perLand* in Abbildung 4.12 zu sehen ist. Bei sehr vielen Standardwerten kann es daher durchaus passieren, dass diese Zeile wie ein vorhandener Datensatz wirkt.

Navigieren in den vorhandenen Datensätzen

Die tatsächliche Anzahl der Datensätze hingegen lässt sich ganz unten ablesen, in Abbildung 4.13 steht der Cursor also offensichtlich im letzten von insgesamt vier Datensätzen. Die höchste *perID* mag anfangs noch ungefähr mit der Datensatzanzahl übereinstimmen, aber gelöschte oder nicht gespeicherte Datensätze hinterlassen Lücken innerhalb der Nummern. Eine zuverlässige Angabe liefert also nur die Angabe in den Navigationsschaltflächen unten.

*Abbildung 4.13: **Die Navigationsschaltflächen und das Suchfeld (unten) für eine Tabelle***

Um einen Datensatz schnell zu finden, gibt es bei den Navigationsschaltflächen das *Suchen*-Feld. Probieren Sie einmal aus, was passiert, wenn Sie dort nacheinander die Buchstaben ung eingeben:

1. Mit u wird der erste Inhalt eines beliebigen Felds gesucht, der ein u enthält. Das ist in diesem Fall die Hauptstraße 99, die direkt markiert wird.
2. Mit un wird anschließend zum fiktiven Ort Hierunddort gesprungen.
3. Bei ung muss dann Prüfungshausen angezeigt werden.

Damit haben Sie schon den wichtigsten Umgang mit einer Tabelle kennen gelernt. Bevor Sie weitere Details sehen, möchte ich Ihnen ein paar Fähigkeiten von Abfragen vorstellen. Schließlich sind Tabellen ja vor allem zum Speichern von Daten da, während erst in Abfragen die Daten strukturiert werden.

4.2 Daten mit Abfragen strukturieren

Um die Daten von Tabellen gefiltert, sortiert oder anders berechnet anzuzeigen, braucht es in Access Abfragen, genauer gesagt Auswahlabfragen. Diese bilden die Gruppe der Abfragen, bei denen salopp gesagt kein Schaden entstehen kann, denn sie zeigen nur Ergebnisse.

Das Gegenstück dazu sind Aktionsabfragen, die auch Daten schreiben. Mit ihnen werden wir uns jedoch erst später befassen. Wenn also im Moment von Abfragen die Rede ist, dann sind nur Auswahlabfragen gemeint.

1. Schließen Sie bitte die Tabelle und aktivieren Sie die Registerkarte *ERSTELLEN*.
2. Dort finden Sie in der Gruppe Abfragen den Befehl Abfrageentwurf. Beim Anklicken öffnet sich direkt ein neues Fenster für die Abfrage und das in Abbildung 4.14 sichtbare Dialogfeld zum Hinzufügen der Tabellen.

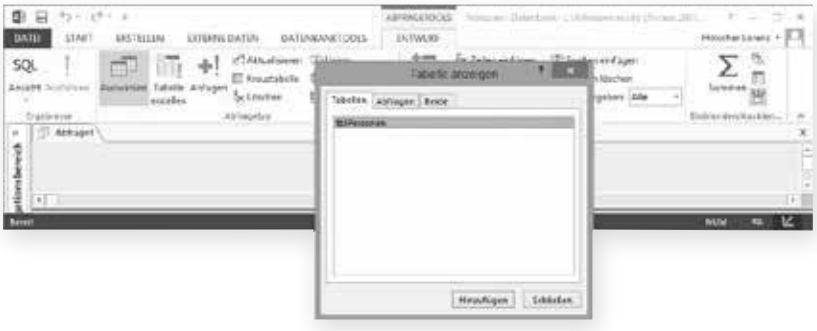


Abbildung 4.14: Erstellen einer neuen Abfrage

3. Darin ist die einzige Tabelle tblPersonen bereits markiert, die Sie bitte mit Hinzufügen in die Abfrage übernehmen. Dann können Sie das Dialogfeld schließen.

Die Methode des Abfrageentwurfs in Access heißt *QBE* oder *Query by example*, also Abfrage durch Beispiel. Dabei sehen Sie im oberen Teil des Abfrageentwurfs alle Datenquellen, wel-

che sowohl Tabellen als auch andere Auswahlabfragen sein können. Deren Felder können Sie durch Ziehen des Feldnamens oder durch Doppelklick darauf in den unteren Teil, nämlich das Beispiel für das Ergebnis, übernehmen.

4. Doppelklicken Sie nacheinander auf die Felder perVorname, perNachname, perPLZ und perOrt wie in Abbildung 4.15 sichtbar.

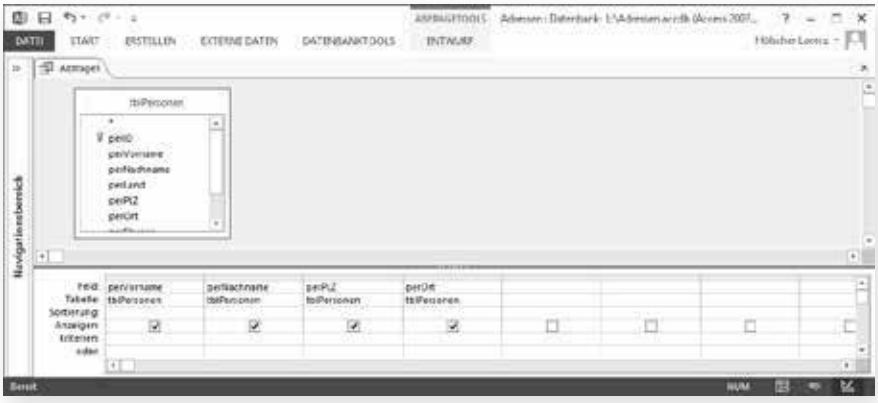


Abbildung 4.15: ***Anzeige der Felder für die neue Abfrage***

5. Um die Abfrage auszuführen und so deren Ergebnisse zu zeigen, muss sie nicht gespeichert werden. Klicken Sie einfach auf das Symbol Ansicht in der automatisch angezeigten *ENTWURF*-Registerkarte, um wie in Abbildung 4.16 die Datenblattansicht zu sehen.

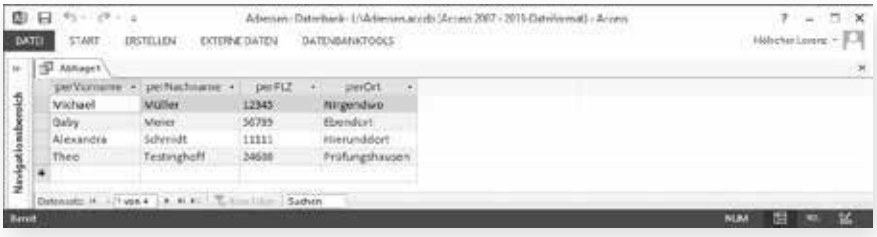


Abbildung 4.16: ***Anzeige der Ergebnisse für die neue Abfrage***

6. Sie können dabei wie in Tabellen auch die Spaltenbreiten durch Ziehen an den Trennlinien zwischen den Spaltenköpfen mit den Feldnamen ändern. Falls Sie von Excel her den Doppelklick an dieser Stelle kennen, optimiert dieser hier ebenfalls die Spaltenbreite.

Allerdings berücksichtigt Access nicht wie in Excel alle Zeilen, sondern nur die auf dem Bildschirm gerade sichtbaren. Schließlich könnten ja noch einige Millionen Zeilen aus einer Datenquelle im langsamen Netzwerk folgen, was viel zu lange dauern würde.

Daten ändern in Abfragen

Was passiert eigentlich, wenn Sie hier den Namen *Müller* in *Müller-Lüdenscheid* ändern? Können und dürfen Sie das? Probieren Sie es einfach mal aus!

Sobald Sie hinter *Müller* klicken und losschreiben, erscheint auch der Schreibstift links im Zeilenkopf. Das ist schon mal ein gutes Zeichen, wirklich gespeichert sind die Daten aber erst, wenn Sie die Zeile ohne Fehlermeldung verlassen konnten. Das klappt auch.

Steht das nicht im Widerspruch zu meiner anfänglichen Behauptung, dass in Tabellen und nur dort Daten gespeichert werden können? Nein, überhaupt nicht. Solange die Abfrage einen Wert genau einem Datensatz in einer Tabelle zuordnen kann, reicht sie alle Änderungen an die passende Tabelle weiter.

Andernfalls – und das betrifft beispielsweise Gruppierungsabfragen, die Summen für mehrere Datensätze zeigen – könnten Sie nicht einmal zu schreiben anfangen. Sobald Access Ihnen also Schreibzugriff auf einen Datensatz gibt, können Sie davon ausgehen, dass Sie ihn auch speichern dürfen.

Hilft es dann vielleicht, die Abfrage ungespeichert zu schließen? Probieren Sie es mit dem *X* am rechten Rand aus und verneinen Sie die Sicherheitsfrage, ob Sie die am Entwurf von *Abfrage1* vorgenommenen Änderungen speichern möchten.

Klappen Sie den Navigationsbereich links (an der Schaltfläche mit den »-Zeichen) aus und doppelklicken Sie dort auf den Tabellennamen *tblPersonen*, um die Daten der Tabelle anzuzeigen. Dort steht jetzt *Müller-Lüdenscheid* im ersten Datensatz. Es hilft also auch nichts, die Abfrage ungespeichert zu verlassen.

Das ist im Grunde auch nicht wirklich überraschend, denn die Abfrage ist ja eine Ansicht der Daten, ein *View,* wie das in englischsprachigen Datenbanken auch genannt wird. Wenn Sie durch ein Fenster sehen, dass Sie von innen mit dem Lichtschalter das Terrassenlicht eingeschaltet haben, verschwindet das Licht ja auch nicht mit der Zerstörung des Fensters.

Das ist durchaus sowohl ein Vorteil als auch ein Nachteil. Als Entwickler können Sie sehr bequem mal eben ein paar Daten filtern und schnell korrigieren. Sie müssen aber auch daran denken, dass Ihre zukünftigen Benutzer mit der gleichen Leichtigkeit versehentlich Daten zerstören können, weil sie vielleicht denken, das sei ja *nur* eine Abfrage und nicht die Tabelle mit den *echten* Daten.

Da es zudem – wie Sie gemerkt haben – Access-typisch keine Aufforderung zum Speichern gibt, fällt die unfreiwillige Datenänderung vermutlich nicht einmal auf. Wer in Access einen geänderten Datensatz nicht speichern will, muss ausdrücklich die Esc -Taste benutzen. Ansonsten speichert Access alles, was nicht gegen Datenintegritätsregeln verstößt.

Eigentlich unterscheiden sich Auswahlabfragen von Aktionsabfragen dadurch, dass sie keine Daten schreiben. Wie Sie gerade gesehen haben, gibt es aber trotzdem die einfache Möglichkeit, als Benutzer Daten per Tastatur zu verändern. Es ist nur eben nicht selbstständig, vollautomatisch und in Bruchteilen von Sekunden für riesige Datenbestände durchgeführt, wie Aktionsabfragen das machen.

Daten sortieren

Im Grunde soll die Auswahlabfrage in unserem Fall ja Daten strukturieren, deswegen erstellen Sie bitte den gleichen Entwurf von eben noch einmal. Es ist sinnvoll, dazu die Tabelle zu schließen, damit nicht unnötig viele Fenster Ihre Rechnerressourcen belasten.

1. Um eine neue Abfrage zu erstellen, müssen Sie zuerst wieder auf der Registerkarte *ERSTELLEN* in der Gruppe Abfragen auf Abfrageentwurf klicken. Übernehmen Sie im Dialogfeld ebenfalls die Tabelle tblPersonen.

2. Dieses Mal sollen die Daten nach Ortsnamen sortiert sein, daher können Sie in der Spalte perOrt in der Zeile Sortierung den Eintrag Aufsteigend wie in Abbildung 4.17 auswählen.

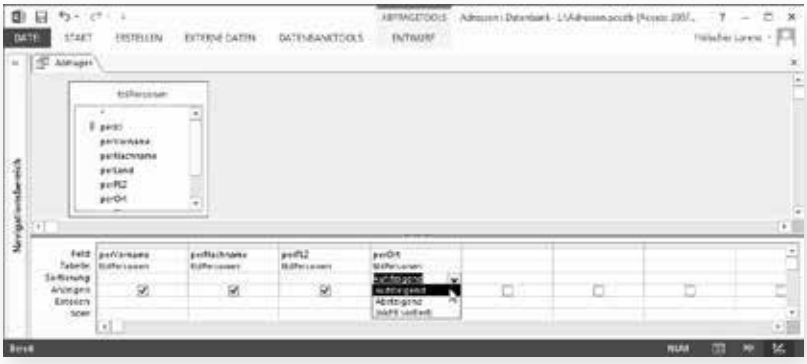


*Abbildung 4.17: **Aufsteigende Sortierung für den Ort im Entwurf***

3. Speichern Sie durch Klick auf die Diskette links oben diesen Abfrageentwurf auf den Namen qryNamenSortiert. Das Präfix qry steht dabei für Query als die englische Bezeichnung von Abfragen.

4. Wechseln Sie nun in die Datenblattansicht (siehe Abbildung 4.18), um das Ergebnis zu überprüfen.

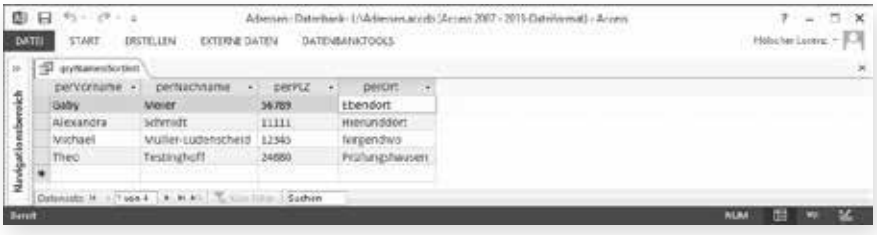


*Abbildung 4.18: **Aufsteigende Sortierung für den Ort im Ergebnis***

Nach mehreren Kriterien sortieren

Im nächsten Schritt soll eine mehrstufige Sortierung durchgeführt werden, das heißt, zuerst soll nach Orten und dann innerhalb eines Ortes nach Nachnamen sortiert werden.

1. Bitte tragen Sie daher noch ein paar Adressen aus bereits vorhandenen Orten ein wie in Abbildung 4.19. Sie können das direkt in dieser Abfrage machen, denn die Daten werden ja an die Tabelle weitergegeben.

So tragen Sie wiederholte Werte ein

Wenn in einem Feld eines Datensatzes der Wert aus dem direkt davor stehenden Datensatz noch einmal benötigt wird, können Sie diesen mit Strg + # übernehmen lassen. Dabei muss er nicht physikalisch in der Tabelle direkt davor stehen, sondern kann wie hier auch in Abfragen durch Filtern oder Sortieren dahin gelangt sein.

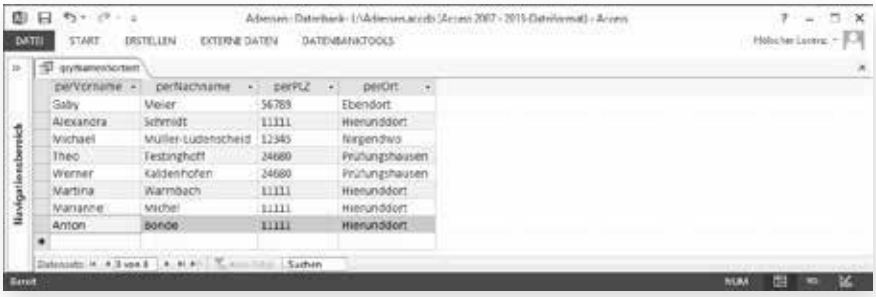


Abbildung 4.19: Zusätzliche Datenzeilen für eine mehrstufige Sortierung

2. Um die mehrstufige Sortierung einzurichten, müssen Sie mit dem Symbol Ansicht ganz links in der Registerkarte *START* wieder in die Entwurfsansicht wechseln. Tragen Sie für das Feld perNachname wie in Abbildung 4.20 ebenfalls eine aufsteigende Sortierung ein.

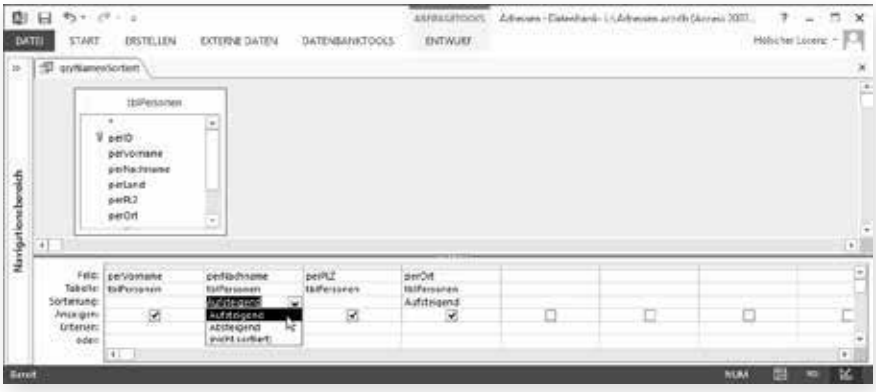


Abbildung 4.20: Eine mehrstufige Sortierung im Entwurf

3. Wenn Sie nun allerdings im Datenblatt das Ergebnis prüfen, werden Sie feststellen, dass keineswegs innerhalb des gleichen Ortes nach den jeweiligen Nachnamen sortiert worden ist, wie Abbildung 4.21 zeigt.

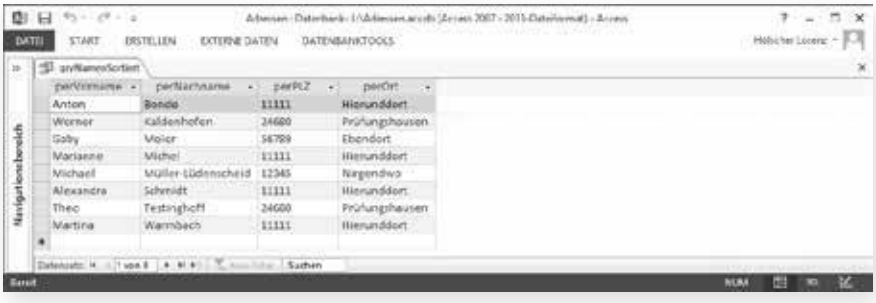


Abbildung 4.21: Die Datensätze sind nicht wie erwartet zuerst nach Ort und dann nach Nachname sortiert

4. Das liegt daran, dass Access natürlich nicht die zeitliche Reihenfolge berücksichtigen kann, in der Sie die Sortierungen im Entwurf eingegeben haben. Vielmehr werden Sortierungen immer von links aus berücksichtigt. Das Feld perOrt muss also weiter links als perNachname stehen, wenn es vorrangig sortiert werden soll.

Felder im Abfrageentwurf verschieben

Um ein Feld im Abfrageentwurf an eine andere Position zu schieben, müssen Sie es zuerst markieren. Der Spaltenkopf oberhalb des Feldnamens ist leider besonders klein und daher schwer zu markieren.

Der Mauszeiger wird im Spaltenkopf als schwarzer Südpfeil dargestellt. Sobald Sie diesen sehen, klicken Sie. Dann wird die ganze Spalte schwarz markiert. Jetzt müssen Sie die Maus wieder loslassen und mit dem nun angezeigten weißen NordWest-Pfeil-Mauszeiger den Spaltenkopf erneut nehmen und gedrückt an die neue Position ziehen.

Während Sie den Spaltenkopf bewegen, springt eine senkrechte schwarze Linie an die zukünftige Position zwischen zwei anderen Spalten. An der richtigen Stelle lassen Sie dann los.

5 Damit der inhaltliche Zusammenhang nicht gestört wird und Vor- und Nachname weiter nebeneinander stehen, wird der Nachname einfach ein zweites Mal per Doppelklick in den unteren Bereich übernommen und steht nun am Ende hinter perOrt.

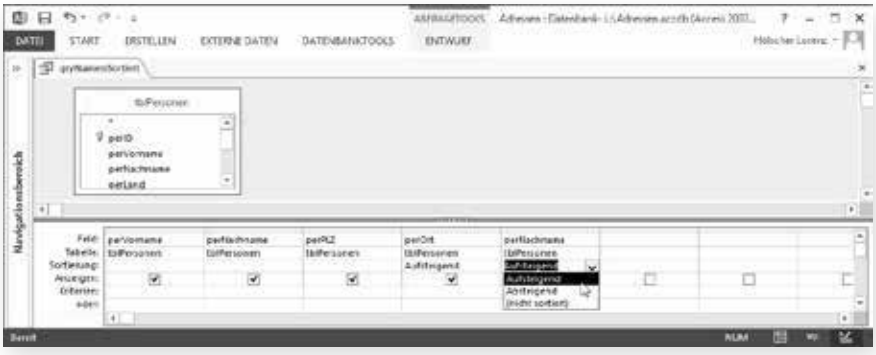
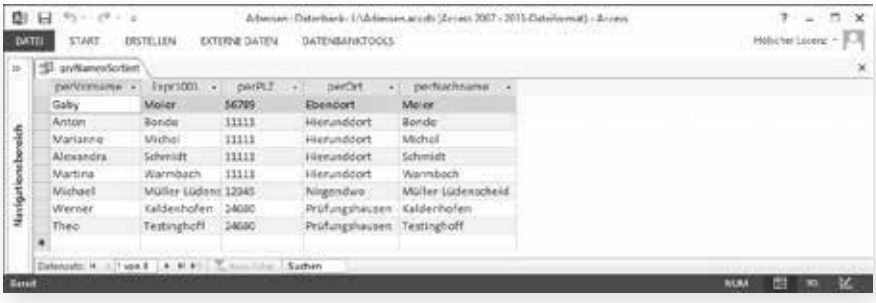


*Abbildung 4.22: **Ändern der Sortierreihenfolge***

6 Abbildung 4.22 zeigt die neue Sortierung. Vergessen Sie nicht, die bisherige Sortierung vom ersten perNachname-Feld wieder auf (nicht sortiert) zu stellen, sonst würde diese ja vorrangig behandelt.



perVorname	Expr1001	perPLZ	perOrt	perNachname
Gaby	Meier	56789	Ebendort	Meier
Anton	Bonde	11111	Hierunddort	Bonde
Marianne	Michel	11111	Hierunddort	Michel
Alexandra	Schmidt	11111	Hierunddort	Schmidt
Martina	Warmbach	11111	Hierunddort	Warmbach
Michael	Müller Lüdens	12345	Ningendwo	Müller Lüdenscheid
Werner	Kalderhofen	24680	Prüfungshausen	Kalderhofen
Theo	Testinghoff	24680	Prüfungshausen	Testinghoff

*Abbildung 4.23: **Diesmal ist die Sortierung inhaltlich korrekt: zuerst nach Orten, dann nach Nachnamen***

Sortierung und Anzeige unterscheiden

Wenn Sie Abbildung 4.23 betrachten, werden Sie feststellen, dass zwar inhaltlich alles korrekt ist, denn es wird tatsächlich zuerst nach Orten und erst anschließend nach Nachnamen sortiert. Allerdings darf es nicht zwei gleichnamige Felder in einem Datenblatt geben, deswegen hat Access sozusagen die Notbremse gezogen und das erste *perNachname*-Feld in *Expr1001* umbenannt. Außerdem ist der wiederholte Nachname am Ende der Tabelle mindestens überflüssig.

Die Lösung besteht darin, im Abfrageentwurf zwischen Feldern zur Anzeige und solchen zur Sortierung zu unterscheiden. Wechseln Sie dazu wieder zurück in die Entwurfsansicht und entfernen Sie im letzten *perNachname*-Feld das Häkchen in der *Anzeigen*-Zeile.

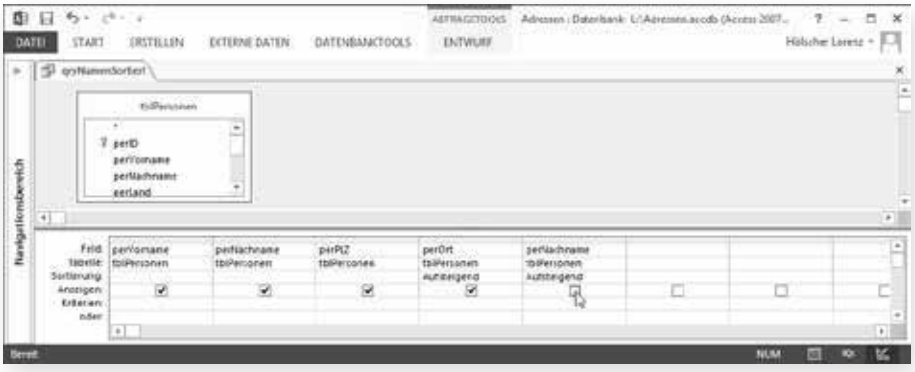


Abbildung 4.24: Entfernen Sie hier das Anzeigen-Häkchen

Jetzt ist auch optisch alles in Ordnung, das *perNachname*-Feld wird im Datenblatt nur einmal vor dem Ortsnamen angezeigt, aber trotzdem nach diesem sortiert.

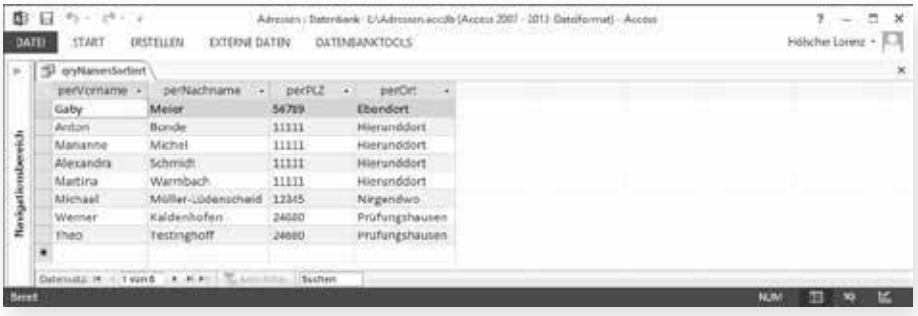


Abbildung 4.25: Diesmal ist die Sortierung inhaltlich und optisch korrekt

Datensätze filtern

Als Nächstes sollen nur die gefilterten Daten für einen einzigen Ort dargestellt werden.

1. Speichern Sie die Abfrage daher unter dem neuen Namen qryNamenGefiltert. Während das Speichern direkt ganz oben mit dem Diskettensymbol möglich ist, finden Sie den Befehl *Speichern unter* in der Backstage-Ansicht, die Sie über die Registerkarte *DATEI* öffnen. Wählen Sie dort *Objekt speichern als* und klicken auf die Schaltfläche *Speichern unter*, um eine Kopie der Abfrage unter einem neuen Namen zu speichern.

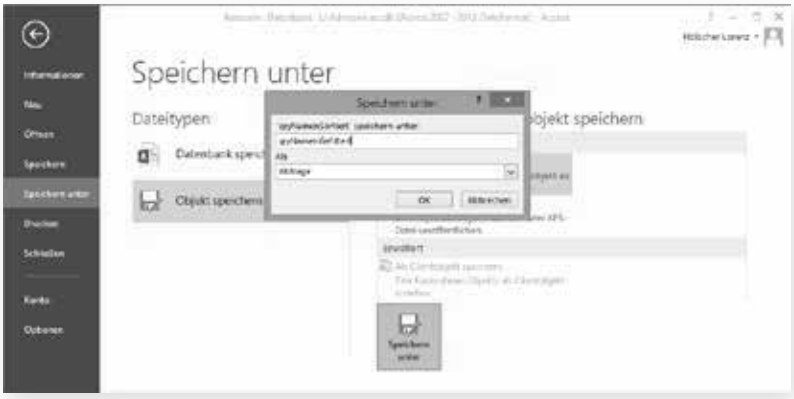


Abbildung 4.26: Speichern einer Abfrage unter einem neuen Namen

2. Entfernen Sie in dieser Abfrage in der Entwurfsansicht nun das letzte perNachname-Feld, indem Sie dessen Spaltenkopf anklicken (der Mauszeiger ändert sich in einen Süd-Pfeil). Dann wird die gesamte Spalte schwarz gefärbt und Sie können anschließend die Entf-Taste drücken.

3. Stellen Sie außerdem bitte die Sortierung von perOrt wieder auf (Nicht sortiert).

Logische Verknüpfung

Damit nur die Datensätze angezeigt werden, die Adressen aus *Hierunddort* enthalten, tragen Sie diesen Ort in der *perOrt*-Spalte in der *Kriterien*-Zeile ein. Die Groß-/Kleinschreibung wird dabei nicht berücksichtigt.

Sobald Sie diese Eingabe mit der ↵-Taste bestätigen, ergänzt Access die fehlenden Gänsefüßchen, denn es ist ein Text-Datentyp. Jetzt können Sie die Abfrage in der Datenblattansicht ausführen lassen und sehen wie gewünscht nur die vier Datensätze aus *Hierunddort*.

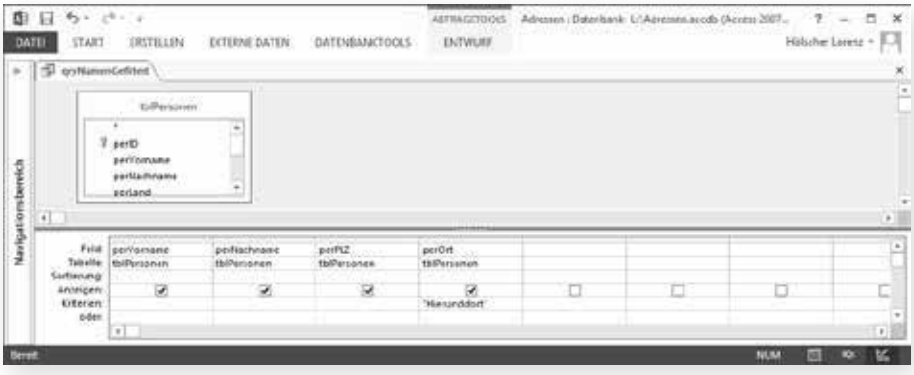


Abbildung 4.27: Filtern Sie die Orte durch Eingabe eines Kriteriums

Natürlich können Sie auch Adressen aus *Hierunddort* und *Prüfungshausen* gemeinsam anzeigen lassen. Da führt Sie aber die deutsche Sprache ein wenig in die Irre. Wenn Sie beispielsweise behaupten, dass Sie morgens mit Bus und Bahn zur Arbeit gekommen seien, dann mag das ökologisch richtig sein. Aber logisch richtig ist es nicht, denn das »und« würde die Gleichzeitigkeit beider Nutzungen verlangen. Der Bus müsste also in der Bahn gefahren sein.

Tatsächlich sind Sie *entweder* mit dem Bus *oder* mit der Bahn gefahren. Unser sprachliches »und« meint also häufig ein logisches »oder«. Deswegen brauchen Sie auch hier ausdrücklich nicht die Adressen aus *Hierunddort* und *Prüfungshausen*, sondern diejenigen aus *Hierunddort* oder *Prüfungshausen.*

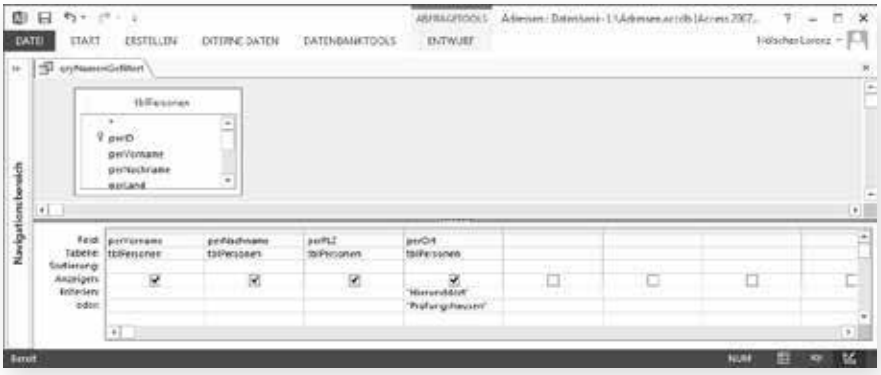


*Abbildung 4.28: **Zwei Kriterien untereinander stehen für ein logisches Oder***

Dazu geben Sie im Entwurf wie in Abbildung 4.28 unterhalb von *Hierunddort* in der nächsten Zeile *Prüfungshausen* ein. Wie Sie in der Datenblattansicht sehen, werden korrekt alle sechs Datensätze aus den beiden Orten angezeigt.

Es hätte übrigens noch eine zweite Art gegeben, das logische *oder* im Entwurf zu schreiben. Das werden Sie sehen, wenn Sie diese Abfrage speichern, schließen und erneut öffnen. Anstatt beide Orte untereinander in zwei Kriterien-Zeilen zu schreiben, hat Access diese wie in Abbildung 4.29 im gleichen Feld durch das Schlüsselwort *Oder* verknüpft.

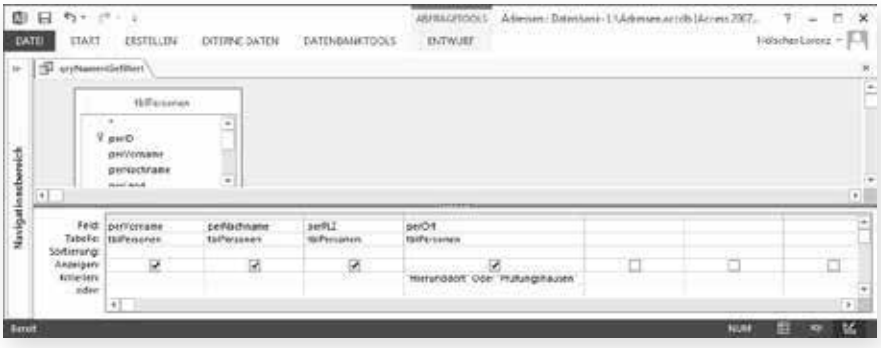


*Abbildung 4.29: **Zwei Kriterien durch Oder im gleichen Feld verbunden***

Damit haben Sie auch schon die Lösung, wenn Sie tatsächlich einmal ein logisches *Und* benötigen. Auch das ist ein Schlüsselwort und kann statt *Oder* eingesetzt werden. Wir werden das im Laufe der Arbeit natürlich auch benutzen.

In diesem Fall würden Sie allerdings feststellen, dass für das Kriterium *„Hierunddort" Und „Prüfungshausen"* gar kein Datensatz als Ergebnis erscheint. Das ist korrekt, denn kein Datensatz würde die Bedingung erfüllen, dass der Ort gleichzeitig *Hierunddort* und *Prüfungshausen* ist.

Damit haben Sie alle wesentlichen Elemente für die Daten einer Access-Datenbank kennen gelernt: Sie können eine Tabelle erstellen und mit einer Abfrage sortieren und filtern.

Übungen zu diesem Kapitel 4.3

In diesem Abschnitt finden Sie einige Übungen zu diesem Kapitel. Die richtigen Antworten finden Sie am Ende des Buchs.

Übung 4.1

Entwerfen Sie bitte eine Datenbank mit einer Tabelle, in der Einnahmen und Ausgaben für eine kleine Firma verwaltet werden können. Zu der Angabe, was gekauft wurde, soll auch vermerkt werden können, wer es gekauft hat. Außerdem ist eine Datumsangabe wünschenswert.

Übung 4.2

Sortieren Sie alle Eintragungen der Tabelle so, dass zuerst nach Datum und (für gleiche Datumswerte) nach Person aufsteigend sortiert wird.

Übung 4.3

Filtern Sie nur die Datensätze für eine bestimmte Person heraus. Wenn das klappt, filtern Sie nach zwei Personen (beachten Sie den Unterschied zwischen *Und* und *Oder!).*