

ACCESS Objekt 2: Abfragen


Während in Tabellen die Daten gespeichert werden, sind Abfragen für die Darstellung und Auswertung der Daten zuständig.




Das Ergebnis einer Abfrage sieht wie eine Tabelle aus. Aber eine Abfrage speichert keine Daten, sondern bezieht ihre Daten stets aus einer oder mehreren Tabellen. Trotzdem kann man meistens in einer Abfrage Daten eingeben und ändern. Das wird dann in der zugrunde liegenden Tabelle gespeichert. Wir werden im Folgenden auch Abfragen kennen lernen, bei denen Access nicht wissen kann, wohin genau die Daten gespeichert werden sollen. Dann ist die Dateneingabe automatisch gesperrt.

Datenbanken nutzen für Abfragen eine eigene Sprache namens **SQL** (Structured Query Language). Mit **Access** kann man Abfragen in der 'Entwurfsansicht' komfortabel grafisch generieren, ohne SQL lernen zu müssen. Natürlich kann man, wenn man mit SQL vertraut ist, direkt damit arbeiten, oder beide Arbeitsweisen mischen.

Nachfolgend wird dargestellt, wie Abfragen in der Entwurfsansicht erstellt werden. Informativ wird auch der SQL-Code in einer vereinfachten und leichter verständlichen Form dargestellt. **Access** muss den Code ja automatisch generieren und legt daher im Zweifel lieber etwas zuviel Text an. Durch dieses 'Mehr' an Text wird die Abfrage weder besser noch schlechter, schneller oder langsamer als die hier verwendete Form.

1. Einfache Abfragen

Wie alle Datenbankobjekte, werden in Access auch neue Abfragen mit  **Neu** angelegt. Das sollte auch hier wieder über die **Entwurfsansicht** passieren. Einem Anfänger helfen auch hier die Assistenten nicht, die Funktionsweise einer Datenbank zu lernen.

Bei vorhandenen Abfragen kommt man mit  **Entwurf** wieder zur Entwurfsansicht und mit  **Öffnen** zur Datenblattansicht. Außerdem gibt es in einer geöffneten Abfrage einen Button **SQL**, mit dem man zur SQL-Ansicht gelangt. Beim Erstellen einer Abfrage öffnet sich das Fenster  **Tabelle anzeigen**, mit dem man die als Datenquelle(n) benötigten Tabellen oder Abfragen hinzufügen kann. Dieses Fenster kann man natürlich auch nachträglich öffnen.

Im oberen Teil des Entwurfsfensters sieht man die ausgewählten Tabellen und Abfragen. Die Darstellung kennen wir schon vom Beziehungsfenster. Neu ist nur das Sternchen vor dem ersten Feldnamen. Im Folgenden beschäftigen wir uns mit der Tabelle *tabOrte*, die wir ja schon aus früheren Beispielen kennen.



Im unteren Teil des Fensters wird die eigentliche Abfrage erstellt.

Feld:		
Tabelle:		
Sortierung:		
Anzeigen:	<input type="checkbox"/>	<input type="checkbox"/>
Kriterien:		
oder:		

Klickt man in die Zeile **Feld**, öffnet sich eine Auswahlliste mit allen Feldnamen, die in den Datenquellen vorhanden sind. Wir wählen jetzt einmal txtOrt aus (Man könnte auch aus der Feldliste im oberen Teil des Fensters einen Feldnamen herunterziehen).

Feld:	txtOrt	
Tabelle:	tabOrte	
Sortierung:		
Anzeigen:	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Kriterien:		
oder:		

Das Feld **Tabelle** wurde automatisch gefüllt und in **Anzeigen** automatisch ein Häkchen gesetzt. Schon haben wir eine vollständige Abfrage erstellt, die wir ausführen können (im Menü/Ribbon **Ansicht/Datenblattansicht** auswählen).

Abfrage1 : Auswahlabfrage	
txtOrt	
München	
Bamberg	
Berlin	
Stuttgart	
Bayreuth	

Wie wir sehen, wird das Feld *txtOrt* aus der Tabelle *tabOrte* angezeigt, und zwar so, als wenn *tabOrte* nur aus diesem Feld bestünde.

Probieren wir einmal folgende Einstellung:

Feld:	txtOrte	IDOrt
Tabelle:	tabOrte	tabOrte
Sortierung:		
Anzeigen:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Kriterien:		
oder:		

Abfrage1 : Auswahlabfrage	
txtOrt	IDOrt
München	1
Bamberg	2
Berlin	3
Stuttgart	4
Bayreuth	5

Die Reihenfolge der Spalten in der Abfrage ist also abhängig von der Reihenfolge, in der sie in der Entwurfsansicht angeordnet sind.


Zurück in der Entwurfsansicht, wählen wir jetzt in der Zeile **Feld** einmal den Wert **tabOrte**.

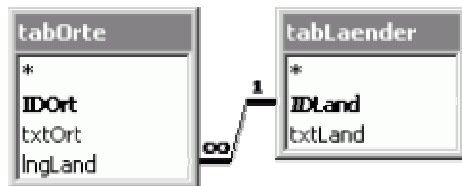
Feld:	tabOrte.*	
Tabelle:	tabOrte	
Sortierung:		
Anzeigen:	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Kriterien:		
oder:		

Abfrage1 : Auswahlabfrage		
IDOrt	txtOrt	IngLand
1	München	2
2	Bamberg	2
3	Berlin	3
4	Stuttgart	1
5	Bayreuth	2

Das Sternchen, das ja auch im oberen Teil des Entwurfsfensters zu sehen ist, ist also einfach eine Abkürzung für 'alle Felder'.

2. Quellenübergreifende Abfragen

Durch die Tabellennormalisierung können wir Daten effektiv speichern. Wie schon erwähnt, sind aber Daten in den Tabellen oft nicht gut lesbar. Daher erstellen wir jetzt eine Abfrage, die dieses Problem löst. Über  **Tabelle anzeigen** fügen wir die Tabellen `tabOrte` und `tabLaender` zu einer Abfrage hinzu. Wir kennen sie ja schon aus früheren Beispielen.



Wie man sieht, wird auch unsere Beziehung automatisch angezeigt. Im unteren Teil des Abfragefensters nehmen wir folgende Einstellung vor:

Feld:	txtOrt	txtLand
Tabelle:	tabOrte	tabLaender
Sortierung:		
Anzeigen:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Kriterien:		
oder:		

Abfrage1 : Auswahlabfrage	
txtOrt	txtLand
München	Bayern
Bamberg	Bayern
Berlin	Berlin
Stuttgart	Baden-Württemberg
Bayreuth	Bayern

Wir haben also Daten aus verschiedenen Tabellen zusammengeführt.

3. Sortieren und Anzeigen

Klickt man in die Zeile **Sortierung**, bekommt man **Aufsteigend**, **Absteigend** oder **(nicht sortiert)** vorgeschlagen. Dies ist soweit selbsterklärend. Wenn die Abfrage nicht sortieren soll (was übrigens performanter ist), richtet sich die Sortierung nach den Schlüsseln der Tabelle.

Nimmt man das Häkchen in der Zeile **Anzeigen** heraus, wird das zugehörige Feld nicht in der Datenblattansicht ausgegeben. Folgendes Beispiel soll den Sinn verdeutlichen:

tabOrte
*
IDOrt
txtOrt
IngLand

Feld:	txtOrt	IngLand	txtOrt
Tabelle:	tabOrte	tabOrte	tabOrte
Sortierung:		Aufsteigend	Aufsteigend
Anzeigen:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Kriterien:			
oder:			

Nun wird zuerst nach der Nummer des Landes, und innerhalb dessen nach dem Ortsnamen sortiert. Trotzdem wird zuerst der Ort angezeigt.

Abfrage1 : Auswahlabfrage		
txtOrt	IngLand	
Stuttgart		1
Bamberg		2
Bayreuth		2
München		2
Berlin		

4. Kriterien

Eine der wichtigsten Fähigkeiten von Abfragen ist, Datensätze nach beliebigen Kriterien bzw. Bedingungen zu filtern.

Hier ein einfaches Beispiel:

tabOrte
*
IDOrt
txtOrt
IngLand

Feld:	txtOrt	
Tabelle:	tabOrte	
Sortierung:		
Anzeigen:	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Kriterien:	"München"	
oder:		

Die Abfrage gibt jetzt nur noch Datensätze aus, bei denen im Feld `txtOrt` 'München' eingetragen ist:

Abfrage1 : Auswahlabfrage	
txtOrt	
München	

Auch im Zusammenhang mit Kriterien ist es manchmal sinnvoll, Felder nicht anzuzeigen:

Feld:	tabOrte.*	txtOrt
Tabelle:	tabOrte	tabOrte
Sortierung:		
Anzeigen:	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Kriterien:		"München"
oder:		

Abfrage1 : Auswahlabfrage

IDOrt	txtOrt	IngLand
1	München	2

Als Kriterium kann man außer einem festen Ausdruck auch **Operatoren** einzugeben:

Feld:	tabOrte.*	IngLand
Tabelle:	tabOrte	tabOrte
Sortierung:		
Anzeigen:	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Kriterien:		>2
oder:		

Alle Datensätze, bei denen in IngLand ein Wert größer als 2 steht, werden ausgegeben;

Abfrage1 : Auswahlabfrage

IDOrt	txtOrt	IngLand
3	Berlin	3

Hier eine Auflistung von Operatoren:

Operatoren		
Operator	Beispiel	Bedeutung
=	=1	gleich
<>	<>1	ungleich
<	<1	kleiner als
>	>1	größer als
<=	<=1	kleiner oder gleich
>=	>=1	größer oder gleich
Zwischen ... und	Zwischen 1 und 3	Zwischen zwei Werten (jeweils einschließlich)

Ferner gibt es den **Null-Operator**, um nach leeren Feldern zu suchen (nicht zu verwechseln mit einer **leeren Zeichenfolge** oder dem Zahlenwert **0**: Um nach einer 0 zu suchen, hieße das Kriterium '=0', um nach einem leeren Feld zu suchen, würde es 'Ist Null' heißen),

Operatoren funktionieren auch für Textfelder:

Feld:	txtOrt	
Tabelle:	tabOrte	
Sortierung:		
Anzeigen:	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Kriterien:		>"C"
oder:		

Nun erhält man alle Ortsnamen, die alphabetisch nach "C" kommen:

Abfrage1 : Auswahlabfrage	
txtOrt	
München	
Stuttgart	

Für Textfelder gibt es noch den **Wie-Operator**. Mit ihm können **Platzhalterzeichen** verwendet werden, um alle Elemente zu finden, die einem Muster entsprechen:

Feld:	tabOrte.*	txtOrt
Tabelle:	tabOrte	tabOrte
Sortierung:		
Anzeigen:	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Kriterien:		Wie "B*"
oder:		

Die Abfrage gibt jetzt alle Datensätze aus, bei denen der Ortsname mit 'B' beginnt:

Abfrage1 : Auswahlabfrage		
IDOrt	txtOrt	IngLand
2	Bamberg	2
5	Bayreuth	2
3	Berlin	3

Folgende Platzhalterzeichen sind möglich:

Platzhalterzeichen			
Symbol	Beispiel	Ergebnis	Verwendung
*	Wie "*er"	findet Maier, Müller, Junker	Entspricht einer beliebigen Anzahl Zeichen
?	Wie "Ma?er"	findet Maier, Majer und Mayer	Entspricht einem beliebigen einzelnen Zeichen
#	Wie "1#3"	findet 103, 113, 123	Entspricht einer beliebigen einzelnen Ziffer im Text
[]	Wie	findet Maier und Mayer, aber nicht	Entspricht einem einzelnen Zeichen innerhalb der eckigen Klammern
!	Wie	findet Majer, aber nicht Maier oder	Entspricht einem einzelnen, beliebigen, nicht aufgelisteten Zeichen
-	Wie "b[a-c]d"	findet bad, bbd und bcd	Entspricht einem einzelnen, beliebigen Zeichen innerhalb des ange-

Es können auch manuell Kriterien (**Parameterwerte**) an eine Abfrage übergeben werden:

Feld:	tabOrte.*	txtOrt
Tabelle:	tabOrte	tabOrte
Sortierung:		
Anzeigen:	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Kriterien:		[Geben Sie einen Ort ein]
oder:		

Führt man diese Abfrage aus, erscheint zunächst eine Eingabeaufforderung, durch die man einen Wert

Parameterwert eingeben
Geben Sie einen Ort ein
<input type="text"/>
OK Abbrechen

an die Abfrage übergeben kann. Interessant werden Abfrageparameter aber erst im Zusammenspiel mit Formularen, da so die Standardeingabeaufforderung durch ein individuelles Formular ersetzt wird:

Feld:	tabOrte.*	txtOrt
Tabelle:	tabOrte	tabOrte
Sortierung:		
Anzeigen:	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Kriterien:		[Forms]![Formularname]![Steuerelementname]
oder:		

5. Mehrere Kriterien

Natürlich kann man beliebig viele Kriterien eingeben, die dann mit **und** bzw. **oder** zueinander in Bezug stehen. Die folgende Einstellung gibt alle Ortsnamen, die mit 'S' oder 'M' beginnen:

tabOrte
*
IDOrt
txtOrt
IngLand

Feld:	tabOrte.*	txtOrt
Tabelle:	tabOrte	tabOrte
Sortierung:		
Anzeigen:	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Kriterien:		"S*"
oder:		"M*"

Abfrage1 : Auswahlabfrage
txtOrt
München
Stuttgart

Jetzt die Kriterien für alle Datensätze, die mit 'B' beginnen und in Bayern liegen:

Feld:	tabOrte.*	txtOrt	IngLand
Tabelle:	tabOrte	tabOrte	tabOrte
Sortierung:			
Anzeigen:	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Kriterien:		"B*"	2
oder:			

Abfrage1 : Auswahlabfrage		
IDOrt	txtOrt	IngLand
2	Bamberg	2
5	Bayreuth	2

Es ist auch möglich, mit **Nicht** eine Bedingung zu verneinen:

Feld:	tabOrte.*	txtOrt
Tabelle:	tabOrte	tabOrte
Sortierung:		
Anzeigen:	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Kriterien:		Nicht "B*"
oder:		

Abfrage1 : Auswahlabfrage		
IDOrt	txtOrt	IngLand
1	München	2
4	Stuttgart	1

6. Abfragefunktionen

Abfragen können auch für verschiedene Berechnungen genutzt werden. Dazu kann mit dem Σ **Funktionen-Button** eine weitere Zeile im unteren Bereich des Entwurfsfensters eingeblendet werden.



Feld:		
Tabelle:		
Funktion:		
Sortierung:		
Anzeigen:	<input type="checkbox"/>	<input type="checkbox"/>
Kriterien:		
oder:		

Klickt man in die Zeile **Funktion**, öffnet sich eine Auswahlliste. Wir wählen zunächst **Gruppierung** und in der Zeile 'Feld' **IngLand**.

Feld:	IngLand	
Tabelle:	tabOrte	
Funktion:	Gruppierung	
Sortierung:		
Anzeigen:	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Kriterien:		
oder:		

Abfrage1 : Auswahlabfrage	
IngLand	
	1
	2
	3

Die Zahl '2' wird nur noch einmal angezeigt, obwohl sie im zugrunde liegenden Tabellenfeld mehrfach vorkommt. Die Gruppierung fasst also gleiche Datensätze zusammen.

Jetzt interessiert uns natürlich, wie viele Datensätze jeweils zusammengefasst wurden:

Feld:	IngLand	IngLand
Tabelle:	tabOrte	tabOrte
Funktion:	Gruppierung	Anzahl
Sortierung:		
Anzeigen:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Kriterien:		
oder:		

Abfrage1 : Auswahlabfrage	
IngLand	AnzahlvonIngLand
1	1
2	3
3	1

In der zugrunde liegenden Tabelle gibt es also dreimal den Eintrag '2', außerdem je einmal eine '1' und eine '3'.

Die übrigen Abfragefunktionen sind selbsterklärend:

Summe
Mittelwert
Min
Max
Anzahl
StAbw
Varianz
Erster Wert
Letzter Wert

7. Berechnungen

Nehmen wir an, wir haben folgende Tabelle:

tabVerkaeufe : Tabelle		
IDVerkauf	datVerkauf	curNettopreis
1	19.10.2010	100
2	20.10.2010	50

Jetzt wollen wir mit einer Abfrage den Bruttobetrag herausfinden. Dazu erstellen wir einfach folgende Abfrage:

tabVerkaeufe	
*	
IDVerkauf	
curNettopreis	

Feld:	tabVerkaeufe.*	Bruttobetrag: [curNettopreis]*1,19
Tabelle:	tabVerkaeufe	
Sortierung:		
Anzeigen:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Kriterien:		
oder:		

Die Abfrage gibt genau das aus, was wir wollen:

Abfrage1 : Auswahlabfrage			
IDVerkauf	datVerkauf	curNettopreis	Bruttobetrag
1	19.10.2010	100	119
2	20.10.2010	50	59,5

8. Aktionsabfragen

Nehmen wir einmal an, ein alter, urbayrischer Wunsch ginge in Erfüllung, und im Rahmen einer Förderalismusreform würde Baden-Württemberg zu Bayern hinzugeschlagen. Zur Erinnerung; wir haben folgende Tabelle:

tabOrte : Tabelle		
IDOrt	txtOrt	IngLand
1	München	2
2	Bamberg	2
3	Berlin	3
4	Stuttgart	1
5	Bayreuth	2

Jetzt müssen also alle Datensätze, die eine '1' im Feld IngLand stehen haben, auf '2' geändert werden. In unserem Beispiel ist das nur ein einziger Datensatz, aber nur, weil wir zu bequem waren, an dieser Stelle Hunderte von Daten einzugeben. Stellen wir uns also eben vor, es wäre nicht nur einer, sondern Hunderte von Datensätzen zugleich zu ändern. Wie dies nun umsetzen?

Zunächst erstellen wir eine Abfrage auf das zu ändernde Feld IngLand, und grenzen es auf die zu ändernden Datensätze ein:

tabOrte
*
IDOrt
txtOrt
IngLand

Feld:	IngLand	
Tabelle:	tabOrte	
Sortierung:		
Anzeigen:	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Kriterien:	1	
oder:		

Abfrage1 : Auswahlabfrage	
IngLand	
	1

Dies sind die zu ändernden Werte. Bisher hatten wir es ausschließlich mit so genannten **Auswahlabfragen** zu tun. Jetzt wird es Zeit für eine **Aktualisierungsabfrage**.

Dazu wählen wir in der Entwurfsansicht den  **Abfragetyp** aus, und ändern die Einstellung auf **Aktualisierungsabfrage**.

Dadurch verändert sich der untere Teil des Entwurfsfensters:

Feld:	IngLand	
Tabelle:	tabOrte	
Aktualisieren:		
Kriterien:	1	
oder:		

--	--	--

Unser Kriterium '1' ist erhalten geblieben, in der Zeile **Aktualisieren** tragen wir jetzt eine '2' ein:

Feld:	IngLand	
Tabelle:	tabOrte	
Aktualisieren:	2	
Kriterien:	1	
oder:		

Wenn wir jetzt einfach wieder in die Datenblattansicht schalten, sehen wir noch immer unsere '1', wie schon zuvor. Obwohl wir jetzt einen anderen SQL-Code haben, hat sich in der Datenblattansicht (außer der Überschrift) nichts verändert.

Abfrage1 : Aktualisierungsabfrage	
IngLand	
	1

Jetzt probieren wir den Button  **Ausführen** aus:

Access	
<p>Sie beabsichtigen, 1 Zeile(n) zu aktualisieren.</p> <p>Sobald Sie auf 'Ja' geklickt haben, können Sie die Änderungen nicht mehr mit dem Befehl 'Rückgängig' zurücknehmen.</p> <p>Sind Sie sicher, dass Sie diese Datensätze aktualisieren möchten?</p>	
Ja	Nein

Wir wählen 'Ja', öffnen danach die Tabelle `tabOrte` und sehen, dass im Feld `IngLand` überall dort, wo bisher eine '1' stand, jetzt eine '2' steht. Unsere Aktualisierungsabfrage hat also genau das gemacht, was wir wollten!

Folgende Arten von Aktionsabfragen gibt es:

- **Tabellenerstellungsabfragen** speichern das Abfrageergebnis in einer neuen Tabelle. Sinnvoll z.B. zum Export in Fremdprogramme oder, um einen aktuellen, sich später ändernden Zustand der Daten festzuhalten.
- **Aktualisierungsabfragen** verändern, wie eben gezeigt, bestehende Daten.
- **Anfügeabfragen** fügen an eine bestehende Tabelle die Datensätze an, die von der Abfrage geliefert werden. Natürlich müssen dafür die Felder in die Tabelle passen.
- **Löschabfragen** löschen unwiederbringlich die Datensätze, die von der Abfrage geliefert werden

9. Kreuztabellenabfragen

Beim folgenden Beispiel gehen wir davon aus, dass die Datenquelle keine Tabelle, sondern eine Abfrage sei. Einerseits um zu zeigen, dass das geht, zum anderen, weil die folgenden Daten in einer Tabelle kaum sinnvoll normalisiert wären:

qryVerkaeufe : Auswahlabfrage				
IDVerkauf	datVerkaufsDatum	txtProdukt	lngStueck	curEinnahme
1	01.02.2010	Kaugummi	10	5,00
2	01.02.2010	Lutscher	15	1,50

3	01.02.2010	Paprikachips	5	10,00
4	02.02.2010	Kaugummi	5	2,50
5	02.02.2010	Paprikachips	5	10,00



Nun wollen wir die Verkäufe eines Tages in einer Zeile sehen. Die Produkte sollen spaltenweise dargestellt werden. Dafür wählen wir den **Abfragetyp** 'Kreuztabelle' aus. Der untere Teil des Abfragefensters verändert sich daraufhin: Der uns schon bekannte Σ **Funktionen-Button** wird aktiviert, eine neue Zeile 'Kreuztabelle' kommt hinzu, und die Zeile 'Anzeigen' verschwindet. Wir füllen dies wie folgt aus:

Feld:	datVerkaufsDatum	txtProdukt	lngStueck
Tabelle:	tabVerkaeufe	tabVerkaeufe	tabVerkaeufe
Funktion:	Gruppierung	Gruppierung	Summe
Kreuztabelle:	Zeilenüberschrift	Spaltenüberschrift	Wert
Sortierung:	Aufsteigend		
Kriterien:			
oder:			

Abfrage1 : Kreuztabellenabfrage			
datVerkaufsDatum	Kaugummi	Lutscher	Paprikachips
01.02.2010	10	15	5
02.02.2010	5		5

Die Abfrage zeigt, wie viele Produkte pro Tag verkauft wurden. Wer das Beispiel am eigenen PC nachverfolgt, sollte auch einmal mit anderen Funktionen in der Spalte 'lngStueck' experimentieren, oder 'lngStueck' durch 'curEinnahme' ersetzen.

10. Tipps & Tricks zu Abfragen

Vor dem Speichern sollte man eine Abfrage mindestens einmal **ausführen**. Access wird sie dabei intern optimieren und die Optimierung zusammen mit der Abfrage speichern. Deswegen sollte man Abfragen auch stets als Abfragen speichern.

In einer Abfrage sollten so wenige **Beziehungen** wie möglich vorkommen. Im Allgemeinen sollte nicht mehr als eine Beziehung zwischen zwei Tabellen vorhanden sein.

Für **Kreuztabellenabfragen** empfehlen sich fixierte Spaltenüberschriften, also die Begrenzung auf benötigte Spalten.

Aktionsabfragen sollten unter VBA weder mit OpenQuery noch mit RunSQL ausgeführt werden, sondern mit DB.Execute.

Werden Abfragen ziemlich komplex, kann es helfen, **die Abfrage zu splitten**, also aus einer Abfrage mehrere zu machen. Häufig stellt sich dabei heraus, dass gleich mehrere Abfragen auf eine 'Basisabfrage' zugreifen können. Beim Splitten der Abfrage sollten folgende Überlegungen greifen:

Die 'Basisabfrage' sollte die Datenmenge eingrenzen. Das geschieht über die Bedingungen (WHERE) sowie die Eingrenzung der benötigten Felder.

Die 'Endabfrage' ist dann zuständig für zeitaufwendige Operationen. Dies sind insbesondere: Gruppieren, Sortieren und berechnete Felder. Auf je weniger Daten diese Abfrage zugreifen muss, desto schneller kann sie arbeiten.