

DATEN MIT SQL ABFRAGEN UND ÄNDERN

Access verfügt über einen leistungsfähigen Abfragegenerator, mit dem sich auch komplexe Aufgabenstellungen lösen lassen. Zusätzlich besteht die Möglichkeit, die Datenbankabfragesprache SQL einzusetzen. An welchen Stellen das sinnvoll ist und welche zusätzlichen Möglichkeiten Sie durch den Einsatz von SQL erhalten, erfahren Sie in diesem Kapitel. Bei SQL (Structured Query Language) handelt es sich um eine so genannte Datenbankabfragesprache, die von allen relationalen Datenbanksystemen verstanden wird.

2.1 Einsatzmöglichkeiten für SQL in Access

Zunächst einmal ist es wichtig zu wissen, dass jede Abfrage, die Sie mit dem grafischen Werkzeug von Access entwickeln, intern in einen SQL-Abfrageausdruck umgewandelt wird. Bei der Ausführung einer Abfrage wird dieser SQL-Befehl an die Jet Datenbankmaschine geschickt. Diese führt die SQL-Anweisungen aus und liefert das Ergebnis an Access zurück.

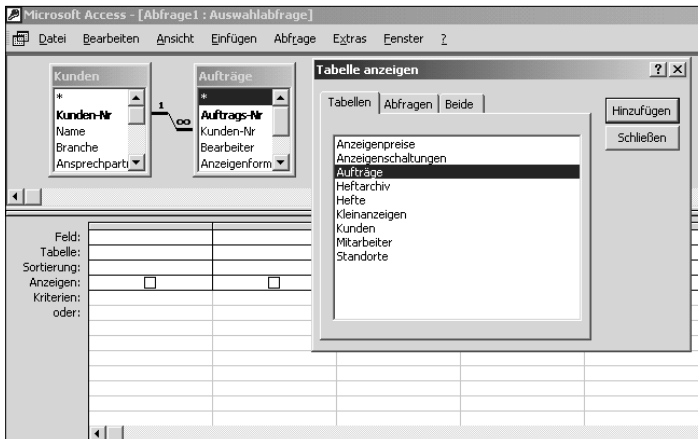
Im folgenden Beispiel lernen Sie kennen, wie Access Ihre Abfrage intern darstellt.

Die SQL-Ansicht einer Abfrage öffnen



Beispiel 8

1. Öffnen Sie die Datenbank *008Stadtlupe* und legen Sie eine neue Auswahlabfrage in der Entwurfsansicht an.
2. Fügen Sie dieser Abfrage die beiden Tabellen *Kunden* und *Aufträge* hinzu.



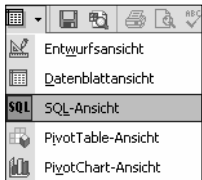
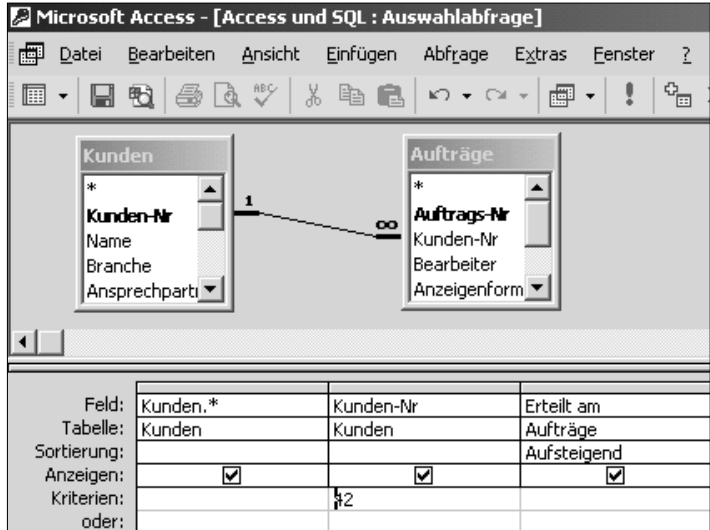
Eine neue Abfrage wird erstellt

3. Erstellen Sie eine Abfragebedingung, in welcher Sie den Kunden mit der Kundennummer 42 suchen.
4. Fügen Sie eine Sortierung nach dem Auftragsdatum hinzu.
5. Führen Sie die Abfrage aus.
6. Wechseln Sie nun von der Tabellenansicht in die SQL-Ansicht. Diese erreichen Sie über das Symbol ganz oben links im Tabellenfenster.



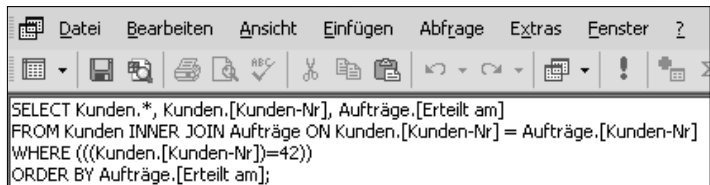
2 DATEN MIT SQL ABFRAGEN UND ÄNDERN

Die Abfrage in der Entwurfsansicht



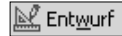
7. In der SQL-Ansicht wird der von Access generierte SQL-Abfrageausdruck dargestellt. Sie können beliebig zwischen der Abfrage-Entwurfsansicht und dem SQL-Fenster wechseln. Es ist auch möglich, den Text in der SQL-Ansicht zu bearbeiten.

Die SQL-Abfrage

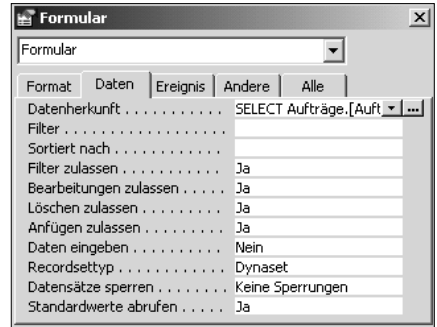


8. Schließen Sie die Abfrage und speichern Sie diese unter dem Namen *Access und SQL*.

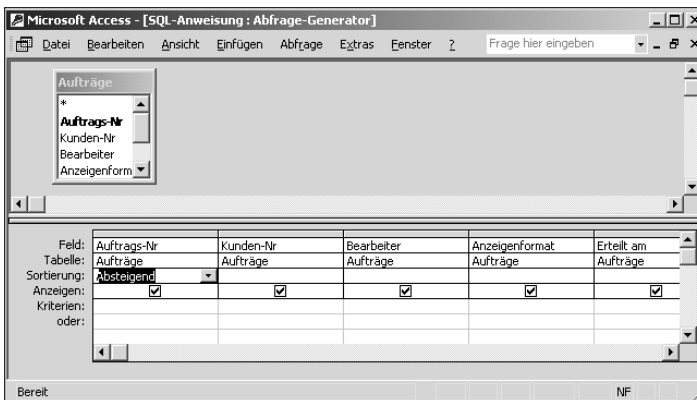
Das nächste Beispiel zeigt Ihnen weitere Stellen in einer Access Datenbank, an denen Sie SQL Ausdrücke entdecken können.



1. Öffnen Sie in der Datenbank 009_010Stadtlupe das Formular *Aufträge* in der Entwurfsansicht.
2. Zeigen Sie über ANSICHT/EIGENSCHAFTEN die Eigenschaften des Formulars an.
3. Im Register DATEN finden Sie die Eigenschaft DATENHERKUNFT. Betrachten Sie den Eintrag dazu. Es handelt sich um einen SQL-Ausdruck: *SELECT Aufträge.[Auftrags-Nr], Aufträge.[Kunden-Nr], Aufträge.Bearbeiter, Aufträge.Anzeigenformat, Aufträge.[Erteilt am], Aufträge.[Bezahlt am], Aufträge.Rabatt FROM Aufträge ORDER BY Aufträge.[Auftrags-Nr];*



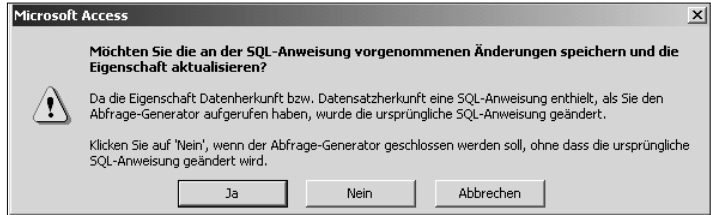
4. Klicken Sie auf die kleine Schaltfläche hinter dem Textfeld. Es öffnet sich der ACCESS-ABFRAGEGENERATOR und Sie können die Abfrage in der bekannten Form sehen.
5. Ändern Sie innerhalb des Abfrage-Entwurfsfensters die Sortierfolge für das Feld AUFTRAGS-NR von *Aufsteigend* in *Absteigend*.
6. Schließen Sie die Abfrage wieder.



Der Abfrage-Generator

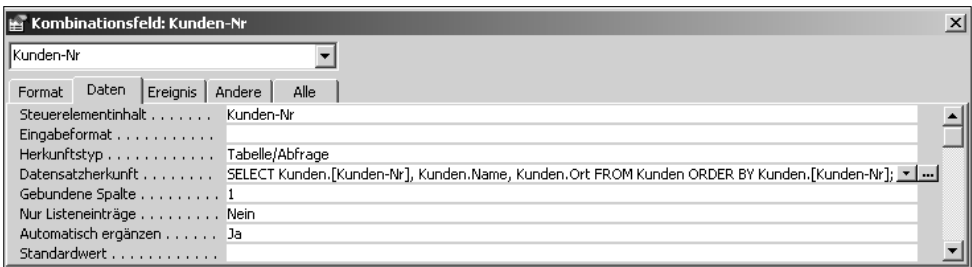
7. Eine Access-Warnmeldung weist Sie darauf hin, dass Sie die ursprüngliche SQL-Anweisung geändert haben. Bestätigen Sie Ihre Änderungen mit JA.
8. Sie gelangen zurück in das Feld DATENHERKUNFT der Formular-eigenschaften, wo Sie den geänderten SQL Ausdruck finden.

2 DATEN MIT SQL ABFRAGEN UND ÄNDERN

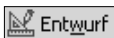


Hinweis →

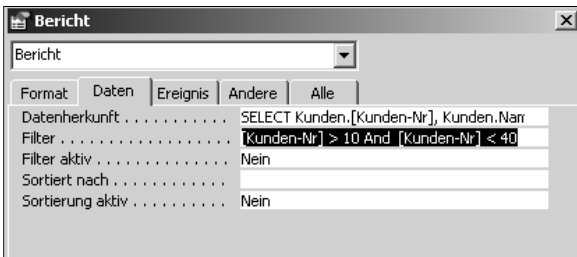
9. Beachten Sie, dass Sie nun auch das Formular selbst einmal speichern müssten, um die geänderte Datenherkunft endgültig zu übernehmen.
10. Überprüfen Sie nun die Eigenschaften des Kombinationsfeldes KUNDEN-NR.
11. Sie finden im Register DATEN die Eigenschaft DATENSATZHERKUNFT . Auch hier sehen Sie als Eintrag einen SQL-Ausdruck.



Die Eigenschaften des Kombinationsfeldes Kunden-Nr.



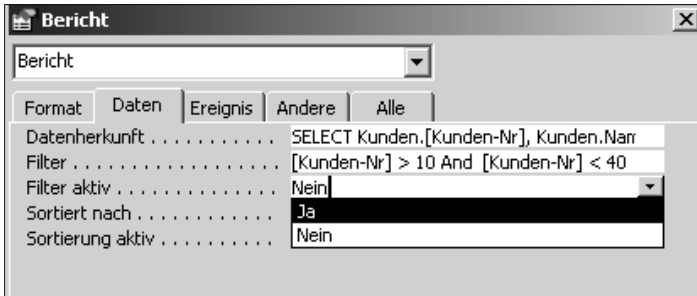
12. Sowohl die Datengrundlage des gesamten Formulars, als auch die Feldliste in einem Kombinationsfeld werden also als SQL-Ausdrücke abgespeichert. Schließen Sie jetzt das Formular, ohne die Änderung zu speichern.



Die Eigenschaften des Berichtes Kundenauswertung

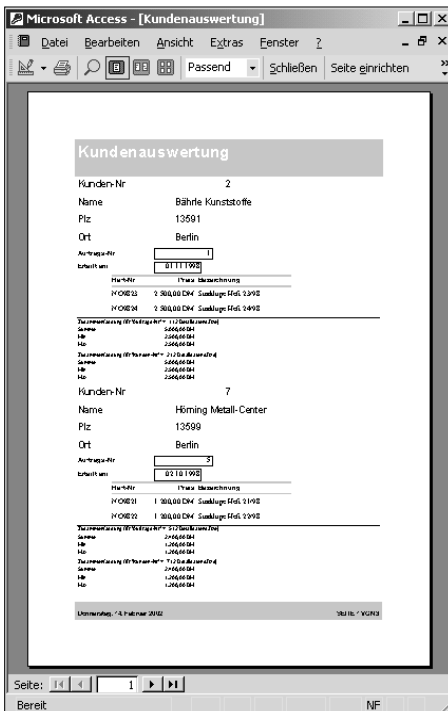
13. Öffnen Sie den Entwurf des Berichtes *Kundenauswertung*.
14. Betrachten Sie nach ANSICHT/ EIGENSCHAFTEN im Register DATEN das Feld FILTER.
15. Dort finden Sie den Eintrag *[Kunden-Nr] > 10 And [Kunden-Nr] < 40*. Sie haben es hier mit einem Teil eines SQL-Ausdrucks zu tun, der eine Einschränkung der durch den Bericht angezeigten Datensätze vornimmt.

16. Stellen Sie die Eigenschaft FILTER AKTIV des Berichtes von *Nein* auf *Ja* um und schließen Sie das Fenster wieder.

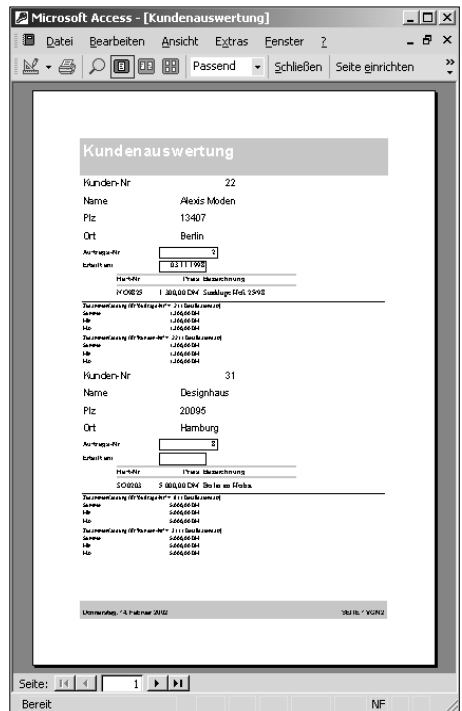


Der FILTER AKTIV wird von *Nein* auf *Ja* umgestellt.

17. Wechseln Sie in die Seitenansicht des Berichtes.
 18. Vergleichen Sie anschließend das Ergebnis mit dem Ergebnis bei deaktiviertem Filter.
 19. Schließen Sie den Bericht, ohne Änderungen zu speichern.



Seite 1 des Berichtes bei ausgeschaltetem Filter



Seite 1 des Berichtes bei aktivem Filter

Sie haben gerade festgestellt, dass es vor allen Dingen die Eigenschaften *Datenherkunft* bzw. *Datensatzherkunft* eines Formulars oder eines Berichtes sind, in welchen man einen SQL-Ausdruck finden kann. Auch in einem Filter kommt zumindest ein Teil eines SQL-Ausdrucks vor.

Darüber hinaus gibt es bei den Access-Abfragen ein paar spezielle Abfragetypen, die Sie ausschließlich mit den Möglichkeiten von SQL verwenden können. Das sind die so genannten UNION-Abfragen und die Unterabfragen. Sie werden für beide im Abschnitt „Fortgeschrittene Abfragetechniken mit SQL“ exemplarische Möglichkeiten finden. Besonders interessant ist der Einsatz von SQL, wenn Access als Frontend für einen Datenbank-Server verwendet wird. Sie können über spezielle Datenschnittstellen (ODBC oder OLE/DB) Abfragen direkt an ORACLE, DB2, SQL-Server oder andere Datenquellen richten. Diese Abfragen werden Pass-Through-Abfragen genannt. Der Einsatz von SQL ermöglicht es Ihnen dabei, in manchen Situationen die Fähigkeiten eines Datenbankservers wesentlich besser zu nutzen, als dies mit einer Standard-Datenbankabfrage möglich wäre.

Eine weitere, höchst interessante Möglichkeit ist der Aufbau eines Client/Server Datenbanksystems unter dem Einsatz der Microsoft Desktop Engine. Access verwendet dabei die Möglichkeiten des Microsoft SQL Servers für sehr leistungsfähige und große Datenbanken. Das Kapitel „Client/Server-Datenbanken mit der Microsoft Desktop Engine und ODBC“ erklärt Ihnen genau, wie das funktioniert. Eine gute Kenntnis von SQL ist die Voraussetzung für die Nutzung dieser Möglichkeit.

2.2 Eine Einführung in die Structured Query Language

SQL – die Structured Query Language - ist eine universelle Datenbanksprache, die von jedem relationalen Datenbankmanagementsystem verstanden werden sollte. Bei SQL handelt es sich um eine der wenigen Programmiersprachen, die von einem Normungsinstitut als Standard verabschiedet wurden. Stand der Dinge ist ANSI SQL-92. Die Zahl 92 steht dabei für das Jahr der Normung. Daran lässt sich erkennen, dass der Standard nicht mehr taufrisch ist. Inzwischen sind leider alle Datenbankhersteller dazu übergegangen, eigene Erweiterungen in ihre Version des SQL einzubauen. Dies führt natürlich dazu, dass der ursprüngliche Gedanke, SQL-Befehle auf identische Art und Weise in unterschiedlichen DBMS (Datenbankmanagementsystemen) nutzen zu können, inzwischen vollständig verwässert ist. Der Umfang der Structured Query Language, den Sie sich in den nachfolgenden Abschnitten erarbeiten, ist jedoch so auf (fast) allen Datenbanksystemen zuhause.

In Access steht Ihnen eine Teilmenge des SQL Standards zur Verfügung. Im Wesentlichen geht es darum, Daten mittels SQL abzufragen oder zu ändern. Das Anlegen von Tabellen oder anderer Datenbankobjekte ist in Access 2002 zwar auch über SQL machbar, spielt aber in der Praxis mehr bei der Programmierung eine Rolle. Dieser Abschnitt vermittelt Ihnen eine Grundlage der Abfragetechniken der Structured Query Language und macht Sie mit deren Anwendung vertraut.

Die folgende Tabelle gibt Ihnen eine Übersicht über die grundlegenden SQL Kommandos.

SQL- Anweisung:	Funktion:
SELECT	Wählt Datensätze aus der Datenbank aus und zeigt sie an.
INSERT	Fügt neue Datensätze in die Datenbank ein.
UPDATE	Ändert Datensätze in der Datenbank.
DELETE	Löscht Datensätze aus der Datenbank.

*Tabelle 1:
Die wichtigsten
SQL-Anweisungen*

Wenn Sie SQL in Ihrer Access-Anwendung einsetzen wollen, haben Sie es an den meisten Stellen mit dem Kommando SELECT zu tun. Aus diesem Grund beginnen wir hier mit der Erläuterung dieses Befehls.

2 DATEN MIT SQL ABFRAGEN UND ÄNDERN

Das Kommando SELECT besteht in den meisten Fällen aus drei Abschnitten. Die einzelnen Abschnitte des Kommandos werden folgendermaßen durch Schlüsselworte gekennzeichnet:

Tabelle 2 :
Die wichtigsten
Befehlstteile des
Kommandos SELECT.

Befehlstteil	Funktion
SELECT	Die SELECT - Anweisung wird zur Anfrage an eine Datenbank und zur Auswahl von Daten, die bestimmte Kriterien zu erfüllen haben, benutzt.
Feldliste	Die Namen, die dem Schlüsselwort SELECT folgen, geben an, welche Spalten in den Ergebnissen berücksichtigt werden sollen. Man kann beliebig viele (existierende) Spaltennamen oder „*“ für sämtliche Spalten einer Tabelle verwenden.
FROM	Der Tabellen-Name, der dem Schlüsselwort FROM folgt, spezifiziert die Tabelle, die für die Anfrage der erwünschten Ergebnisse durchsucht werden soll.
Tabellen	Angabe, aus welchen Tabellen die Daten stammen.
WHERE	Die WHERE Klausel spezifiziert, welche Werte in der Tabelle bzw. welche Zeilen angezeigt bzw. ausgewählt werden sollen.
Bedingung	Einschränkung des Ergebnisses.

Im folgenden Beispiel lernen Sie, Abfragen direkt in SQL zu formulieren.

Beispiel 10



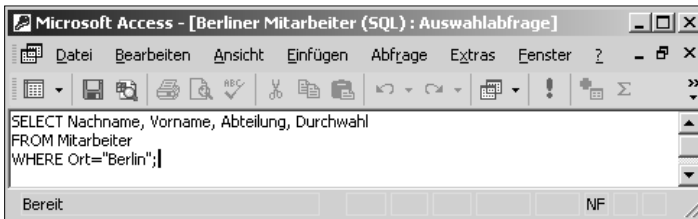
Eine einfache SELECT-Abfrage



1. Die Datenbank 009_010Stadtlupe ist geöffnet.
2. Erstellen Sie eine neue Abfrage in der Entwurfsansicht.
3. Sie möchten der Abfrage keine Tabellen hinzufügen. Klicken Sie im Dialogfenster TABELLE ANZEIGEN auf SCHLIESSEN.

Objekte	Name
Tabellen	Erstellt eine neue Abfrage in der Entwurfsansicht
Abfragen	Erstellt eine Abfrage unter Verwendung des Assistenten
Formulare	Access und SQL
	Alle Kunden und Aufträge (SQL)

4. Wechseln Sie in die SQL-Ansicht der Abfrage.
5. Geben Sie nun als erstes den folgenden Text ein:
SELECT Nachname, Vorname, Abteilung, Durchwahl.
Hiermit legen Sie fest, welche Felder von ACCESS angezeigt werden sollen.
6. Geben Sie nun *FROM Mitarbeiter* ein. Damit geben Sie als Datenquelle die Tabelle *Mitarbeiter* an.
7. Geben Sie nun *WHERE Ort="Berlin"* ein. Sie schränken durch eine Bedingung die angezeigten Datensätze auf den Standort Berlin ein.



Die SQL-Abfrage

8. Beenden Sie die Abfrage mit einem Semikolon.
9. Führen Sie die Abfrage nun aus.
10. Wenn Sie bei der Eingabe keine Fehler gemacht haben, zeigt Ihnen Access die Daten der Berliner Mitarbeiter an.



	Nachname	Vorname	Abteilung	Durchwahl
▶	Sternthal	Alf	Freier Fotograf	
	Späth	Andrea	Journalist	
	Lamprecht	Horst	Anzeigen	25
	Egeyd	Josef	Journalist	
	Aenter	Klaus	Redakteur	
	Heinerle	Karin	Redakteur	
	Müller	Karin	Anzeigen	
	Mayer	Karl	Redaktion Essen und Trinken	12
	Schulze	Klaus	Freier Journalist	

Datensatz: 1 von 17

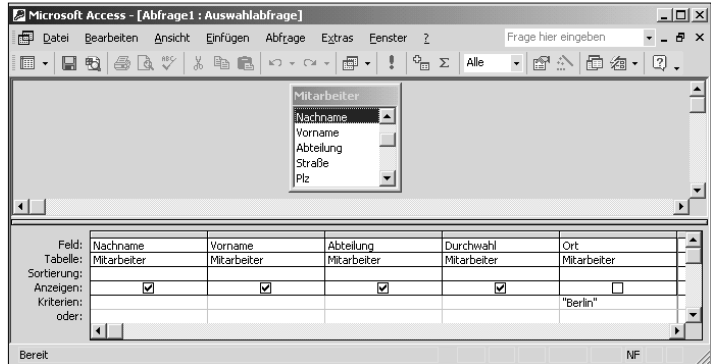
Die Daten der Berliner Mitarbeiter werden angezeigt.

11. Betrachten Sie das Ergebnis nun in der Entwurfsansicht der Abfrage. Access kann Ihren SQL Ausdruck wie eine normale Abfrage darstellen.



2 DATEN MIT SQL ABFRAGEN UND ÄNDERN

Die Abfrage im Abfragegenerator



12. Speichern Sie die Abfrage unter dem Namen *Berliner Mitarbeiter (SQL)* ab.

Natürlich hätten Sie diese Abfrage auch mit dem Abfragegenerator erstellen können. Sie sollten sich durch die direkte Verwendung der SQL-Kommandos allerdings zunächst an die Schreibweise gewöhnen. Im nächsten Abschnitt dieses Lehrgangs lernen Sie Verfahren kennen, die auf der direkten Verwendung von SQL-Ausdrücken beruhen. Es lohnt sich also, deren Verwendung zu üben.

Das nächste Beispiel stellt eine Variante der Feldliste vor.

Beispiel 11



Alle Felder einer Tabelle anzeigen lassen



1. Die Datenbank *011Stadtlupe* ist geöffnet.
2. Legen Sie eine neue Auswahlabfrage an.
3. Geben Sie in der SQL-Ansicht `SELECT * FROM Mitarbeiter;` ein.
4. Führen Sie diese Abfrage aus.
5. Alle Felder und sämtliche Datensätze aus der Mitarbeitertabelle werden angezeigt. Mit anderen Worten: Sie öffnen die komplette Tabelle. Der Stern * teilt Access mit, dass alle Felder angezeigt werden sollen. Durch das Weglassen der WHERE-Klausel werden die Datensätze nicht eingeschränkt.
6. Speichern Sie diese Abfrage unter dem Namen *alle Mitarbeiter (SQL)* ab.

Mitarbeiter-Code	Standort-ID	Nachname	Vorname	Abteilung	Straße	Plz	Ort	Durchwahl	Fax
AS	B	Sternthal	Alf	Freier Fotograf	Holtzendorffstraße	14075	Berlin		
ASP	B	Späth	Andrea	Journalist	Penzberger Str. 34	10779	Berlin		
HL	B	Lamprecht	Horst	Anzeigen	Zimmermannstr. 23	10119	Berlin	25	99
HM	H	Müller	Hans	Vertrieb	Elbchaussee 104	22345	Hamburg		
HS	M	Schmidchen	Heribert	Redakteur	An der Isar 2	80123	München		
JB	B	Bruckner	Jost	Freier Journalist	Roßstr. 76	80456	München		
JE	B	Egeyd	Josef	Journalist	Bergmannstr. 123	10557	Berlin		
JM	M	Mayer	Josph	Freier Fotograf	Promenadenplatz2		München		
KA	B	Aenter	Klaus	Redakteur	Perleberger Str. 11	10559	Berlin		

Die Abfrage wird ausgeführt

Jetzt lernen Sie die WHERE-Klausel kennen. Sie werden viele Ähnlichkeiten zwischen den Access-Abfragebedingungen und den Elementen der WHERE-Klausel feststellen. Die bekannten Abfrageoperatoren tauchen hier unter neuem Namen wieder auf. Auch die logischen Verknüpfungoperatoren sind vorhanden. Die beiden folgenden Tabellen geben Ihnen eine Übersicht:

Abfrageoperator:	Funktion:	Beispiel:
=, >, <, >=, <=	Wertevergleich	WHERE Preis > 500
LIKE	Textvergleich mit Ersetzungszeichen	WHERE [Name] LIKE 'sch*'
BETWEEN	Bereichsvergleich	WHERE Preis BETWEEN 100 AND 200
IN	Vergleich mit einer Liste	WHERE Preis IN (100, 200, 300)
IS NULL	Test, ob das Feld einen Wert besitzt.	WHERE Preis IS NULL
EXISTS	Test, ob eine Unterabfrage einen Wert liefert.	WHERE EXISTS (SELECT * FROM Kunden)

Tabelle 3
Die Abfrageoperatoren

Die Platzhalterzeichen bei Textvergleichen entsprechen im SQL-Modus den in Access verwendeten. Der Vollständigkeit halber seien diese hier aufgezählt:

- * ersetzt ein oder mehrere beliebige Zeichen. „Schul*“ steht also für „Schulze“, „Schultze“ aber auch „Schul“.
- ? ersetzt ganz genau ein Zeichen. „Schul?e“ steht damit für „Schulze“, „Schulte“ aber nicht für „Schule“. Der Buchstabe muss auf jeden Fall vorhanden sein.

→ Durch eckige Klammern [...] kann man eine Anzahl Zeichen festlegen, die an dieser Stelle im Text vorkommen dürfen. „Schul[t,z]e“ ermöglicht damit die Bildung der Worte „Schulze“ und „Schulte“. „Schul[a-z]e“ ist eine weitere Form. In den eckigen Klammern wird ein Bereich von Zeichen angegeben, die verwendet werden dürfen. Hier also Buchstaben und keine Zahlen.

Genau so, wie Sie es von Access her gewohnt sind, können Sie mehrteilige Abfragebedingungen einrichten. Die einzelnen Teile werden durch die folgenden Operatoren miteinander verbunden:

Tabelle 4:
Logische Operatoren

Logischer Operator:	Funktion:
AND	Verknüpft zwei Bedingungen durch ein logisches UND.
OR	Verknüpft zwei Bedingungen durch ein logisches ODER.
NOT	Negiert eine Bedingung.

Insgesamt könnte eine Abfrage wie folgt aussehen:

```
SELECT * FROM Kunden WHERE Ort="Berlin" AND Not Plz Like "12*";
```

Diese Abfrage würde Kunden auswählen, die in Berlin ansässig sind, aber nicht in den Postleitzahlbereich 12... fallen.

Beachten Sie die richtige Reihenfolge der Operatoren. Sicherheitshalber können Sie die Reihenfolge immer durch runde Klammern festlegen:
*SELECT * FROM Kunden WHERE ((Ort="Berlin") AND (Plz Not Like "12*"));*

Diese Abfrage finden Sie unter: *Berliner Kunden ohne PLZ 12 (SQL)*. Beachten Sie dazu auch das folgende Beispiel:

Beispiel 12



Daten mit der WHERE-Klausel auswählen



1. Legen Sie in der Datenbank *012Stadtlupe* eine neue Auswahlabfrage an.



2. Wechseln Sie in die SQL-Ansicht

3. Geben Sie den ersten Teil der Abfrage ein:

```
4. SELECT Nachname, Vorname, [Standort-ID], Abteilung, Extern  
FROM Mitarbeiter
```

5. Es folgt die Auswahlbedingung:
*WHERE [Standort-ID]="B" AND (Extern=Yes OR
 Abteilung="Redakteur");*
6. Führen Sie diese Abfrage aus.
7. Betrachten Sie das Ergebnis. Sie sollten nur Mitarbeiter sehen, die am Standort Berlin arbeiten und entweder freie Mitarbeiter sind oder als Redakteure arbeiten.



	Nachname	Vorname	Standort-ID	Abteilung	Extern
▶	Bruckner	Jost	B	Freier Journalist	<input checked="" type="checkbox"/>
	Aenter	Klaus	B	Redakteur	<input checked="" type="checkbox"/>
	Heinerle	Karin	B	Redakteur	<input type="checkbox"/>
	Schulze	Klaus	B	Freier Journalist	<input checked="" type="checkbox"/>
	Gelp	Renate	B	Freier Journalist	<input checked="" type="checkbox"/>
*					<input type="checkbox"/>

Datensatz: 1 von 5

Mitarbeiter, die in Berlin arbeiten und entweder freie Mitarbeiter oder Redakteure sind.

8. Speichern Sie die Abfrage unter dem Namen *Redakteure und freie Mitarbeiter aus Berlin (SQL)*.
9. Wechseln Sie wieder in die SQL-Ansicht, entfernen Sie die Klammern und führen Sie die Abfrage erneut aus. Sie erhalten im Ergebnis nun die Berliner freien Mitarbeiter und alle Redakteure.

	Nachname	Vorname	Standort-ID	Abteilung	Extern
▶	Schmidchen	Heribert	M	Redakteur	<input type="checkbox"/>
	Bruckner	Jost	B	Freier Journalist	<input checked="" type="checkbox"/>
	Aenter	Klaus	B	Redakteur	<input checked="" type="checkbox"/>
	Friesen	Klaus	H	Redakteur	<input type="checkbox"/>
	Heinerle	Karin	B	Redakteur	<input type="checkbox"/>
	Schulze	Klaus	B	Freier Journalist	<input checked="" type="checkbox"/>
	Gelp	Renate	B	Freier Journalist	<input checked="" type="checkbox"/>
*					<input type="checkbox"/>

Datensatz: 1 von 7

Die freien Mitarbeiter aus Berlin und alle Redakteure

10. Schließen Sie die veränderte Abfrage, ohne sie zu speichern.

Wir kommen nun zum Sortieren des Abfrageergebnisses. In SQL ist dafür die Klausel `ORDER BY` zuständig. Diese steht immer ganz am Ende eines SQL-Kommandos. Sie können ein oder mehrere Felder angeben, nach denen sortiert werden soll, und Sie können zu jedem Feld angeben, ob die Sortierreihenfolge aufsteigend (ASC) oder absteigend (DESC) ist. Da aufsteigende Sortierung die Voreinstellung ist, kann ASC auch weggelassen werden.

Tabelle 5:
Sortierklauseln

Sortierklausel:	Funktion:	Beispiel:
ORDER BY feld ASC	Aufsteigende Sortierung nach dem Feldinhalt.	ORDER BY Ort ASC
ORDER BY feld DESC	Absteigende Sortierung nach dem Feldinhalt.	ORDER BY Ort DESC

`SELECT * FROM Kunden ORDER BY Ort DESC;` gibt also sämtliche Kundendatensätze absteigend sortiert nach dem Ort zurück.

Beispiel 13



Daten mit der ORDER BY-Klausel sortieren



1. Legen Sie in der Datenbank *013Stadtlupe* eine neue Auswahlabfrage an.

2. Wechseln Sie in die SQL-Ansicht.



3. Geben Sie den ersten Teil der Abfrage ein:

`SELECT * FROM Mitarbeiter`

4. Nun folgt die Sortierklausel:

`ORDER BY [Standort-ID], Extern DESC, Nachname;`

5. Führen Sie diese Abfrage aus.

6. Die Mitarbeiterinformationen werden sortiert in der angegebenen Reihenfolge dargestellt. Mit Ausnahme des Feldes `EXTERN` ist die Sortierreihenfolge aufsteigend. Durch das Schlüsselwort `DESC` wird das Feld `EXTERN` absteigend sortiert und daher die externen Mitarbeiter hinter den internen angezeigt.

7. Speichern Sie diese Abfrage unter dem Namen *Mitarbeiter sortiert (SQL)*.

Eine Methode, die man häufig im Zusammenhang mit der Sortierung von Datensätzen sieht, ist die Einschränkung der Ergebnismenge auf eine bestimmte Anzahl von Zeilen. Dadurch können Fragen in der Art „zeige mir die umsatzstärksten Artikel“ beantwortet werden. Auch für die weitere Aufbereitung der Daten ist die Eingrenzung oft sinnvoll. Ein Balkendiagramm mit mehr als 20 Balken macht häufig keinen Sinn und es reicht häufig, die wesentlichen Werte zu sehen.

Das nächste Beispiel demonstriert den Einsatz des TOP-Schlüsselwortes, mit welchem das gerade genannte Ziel erreicht werden kann. Es sollen die zwanzig auflagenstärksten Hefte gezeigt werden.

Das Ergebnis mit TOP einschränken



Beispiel 14

1. Die Datenbank *014Stadtlupe* ist geöffnet.
2. Legen Sie eine neue Abfrage in der SQL Ansicht an.
3. Lassen Sie sich mit dem folgenden SQL-Befehl die Hefte anzeigen:
`SELECT [Heft-Nr], Bezeichnung, Auflage FROM Hefte ORDER BY Auflage DESC;`



Das Ergebnis ist also nach der Auflagenhöhe absteigend sortiert.

Es werden 98 Datensätze angezeigt.

4. Modifizieren Sie nun die SELECT-Klausel durch das TOP-Schlüsselwort: `SELECT TOP 20 [Heft-Nr], Bezeichnung, Auflage FROM Hefte ORDER BY Auflage DESC;` Im Ergebnis werden jetzt nur noch 20 Datensätze angezeigt.
5. Speichern Sie die Abfrage unter dem Namen *AuflagenTop20 (SQL)*.

Heft-Nr	Bezeichnung	Auflage
NO 01-19	Stadtlupe Heft 19/01	110000
NO 00-03	Stadtlupe Heft 3/00	110000
NO 01-03	Stadtlupe Heft 3/01	110000
NO 01-02	Stadtlupe Heft 2/01	110000
NO 01-18	Stadtlupe Heft 18/01	110000
NO 01-17	Stadtlupe Heft 17/01	110000
NO 01-16	Stadtlupe Heft 16/01	110000
NO 01-15	Stadtlupe Heft 15/01	110000
NO 01-14	Stadtlupe Heft 14/01	110000
NO 01-13	Stadtlupe Heft 13/01	110000
NO 01-06	Stadtlupe Heft 6/01	110000
NO 01-04	Stadtlupe Heft 4/01	110000
NO 01-01	Stadtlupe Heft 1/01	110000
NO 01-12	Stadtlupe Heft 12/01	110000
NO 01-05	Stadtlupe Heft 5/01	110000
NO 01-07	Stadtlupe Heft 7/01	110000
NO 01-08	Stadtlupe Heft 8/01	110000
NO 01-09	Stadtlupe Heft 9/01	110000
NO 01-10	Stadtlupe Heft 10/01	110000
NO 01-11	Stadtlupe Heft 11/01	110000
*		0

Datensatz: 1 von 20

Das Ergebnis der Abfrage

Die folgende Technik ist Ihnen sicher aus der Arbeit mit Abfragen in ACCESS bekannt. Es geht um die Vermeidung von Doubletten in einem Abfrageergebnis. Sie lernen im nächsten Beispiel die SQL-Varianten der entsprechenden Abfrage-Eigenschaften kennen.

Beispiel 15



Doubletten mit DISTINCT und DISTINCTROW vermeiden



1. Legen Sie in der Datenbank *015Stadtlupe* eine neue Auswahlabfrage an.
2. Wechseln Sie in die SQL-Ansicht.



3. Erzeugen Sie eine Liste der Kundenorte über den SQL-Befehl:

```
SELECT Plz, Ort  
FROM Kunden  
ORDER BY Plz;
```

4. Überprüfen Sie das Ergebnis. Sie werden feststellen, dass einige Kombinationen aus Postleitzahl und Ort sich wiederholen. Dies soll verhindert werden.
5. Ergänzen Sie den SQL-Ausdruck um das Schlüsselwort DISTINCT. Er sollte jetzt wie folgt lauten:

```
SELECT DISTINCT Plz, Ort  
FROM Kunden  
ORDER BY Plz;
```

In der Datenblattansicht werden Sie keine doppelten Einträge mehr finden.

6. Ändern Sie den SQL-Ausdruck erneut, indem Sie DISTINCT durch DISTINCTROW ersetzen:

```
SELECT DISTINCTROW Plz, Ort  
FROM Kunden  
ORDER BY Plz;
```

7. Führen Sie die Abfrage erneut aus. Sie erhalten nun wieder die gesamte Datenmenge, inklusive Doubletten. Der Unterschied ist folgender: DISTINCT schließt Wiederholungen in den angezeigten Daten aus, während DISTINCTROW die Datensätze insgesamt untersucht. DISTINCTROW wird ignoriert, wenn die Abfrage nur eine Tabelle umfasst.

Plz	Ort
02826	Görlitz
02826	Görlitz
06842	Dessau
06842	Dessau
06844	Dessau
07545	Gera
07545	Gera
07548	Gera
07548	Gera
10115	Berlin

Mit **SELECT** werden *doppelte* Datensätze angezeigt

Plz	Ort
02826	Görlitz
06842	Dessau
06844	Dessau
07545	Gera
07548	Gera
10115	Berlin
10318	Berlin
10437	Berlin
10623	Berlin
12045	Berlin

Mit **SELECT DISTINCT** werden *keine doppelten* Datensätze mehr angezeigt

Plz	Ort
02826	Görlitz
02826	Görlitz
06842	Dessau
06842	Dessau
06844	Dessau
07545	Gera
07545	Gera
07548	Gera
07548	Gera
10115	Berlin

Mit **SELECT DISTINCTROW** werden *wieder doppelte* Datensätze angezeigt

- In der MS JET SQL-Referenz der MS Access-Hilfe erhalten Sie weitere Informationen u. a. zur Verwendung der ALL-, DISTINCT-, DISTINCTROW- und TOP-Prädikate.
- Zeigen Sie die Abfrage in der normalen Entwurfsansicht an. Klicken Sie mit der rechten Maustaste in die freie Fläche und wählen Sie aus dem Kontextmenü **EIGENSCHAFTEN**. Die Eigenschaft *Eindeutige Datensätze* ist auf **Ja** eingestellt. Dies entspricht dem SQL-Ausdruck **DISTINCTROW**.

← *Hinweis*

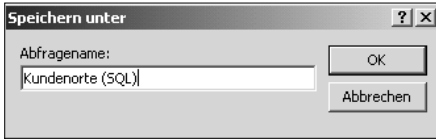
Abfrageeigenschaften	
Allgemein	
Beschreibung	
Standardsicht	Datenblatt
Alle Felder ausgeben	Nein
Spitzenwerte	Alle
Keine Duplikate	Nein
Eindeutige Datensätze	Ja
Ausführungsberechtigungen	Benutzer
Quelldatenbank	(aktuell)
Quellverbindung	
Datensätze sperren	Keine Sperrungen
Recordsettyp	Dynaset
ODBC-Wartezeit	60
Filter	

DISTINCTROW: Die Abfrageeigenschaft *eindeutige* Datensätze ist auf **Ja** eingestellt

Abfrageeigenschaften	
Allgemein	
Beschreibung	
Standardsicht	Datenblatt
Alle Felder ausgeben	Nein
Spitzenwerte	Alle
Keine Duplikate	Ja
Eindeutige Datensätze	Nein
Ausführungsberechtigungen	Benutzer
Quelldatenbank	(aktuell)
Quellverbindung	
Datensätze sperren	Keine Sperrungen
Recordsettyp	Dynaset
ODBC-Wartezeit	60
Filter	

Der SQL-Ausdruck **DISTINCT** entspricht der Abfrageeigenschaft *keine Duplikate*.

- Ändern Sie die Eigenschaft *Keine Duplikate* auf **Ja**. Vergleichen Sie dies mit der SQL-Darstellung der Abfrage. Es wird wieder **DISTINCT** verwendet.



11. Speichern Sie die Abfrage unter dem Name *Kundenorte (SQL)* und schließen Sie diese.

In vielen Fällen werden in Abfragen neben den eigentlichen Tabellenfeldern noch weitere Werte benötigt, die über eine Berechnung zustande kommen. Selbstverständlich ist diese Möglichkeit auch in SQL vorgesehen. Allerdings weicht die Syntax von der üblichen Access Schreibweise ab. In Access wird ein berechnetes Feld ja über die Syntax *Feldname: Ausdruck* angegeben. SQL verwendet dagegen die Form *Ausdruck AS Feldname*. Zunächst wird also der Berechnungs-Ausdruck angegeben, danach folgt das Schlüsselwort *AS* und zum Schluss der Feldname, den Sie vergeben möchten.

In den Ausdrücken selbst können Sie so gut wie alle Möglichkeiten verwenden, die Access bietet. Im nächsten Kapitel können Sie sich genauer über Ausdrücke in Access informieren.

Beispiel 16



Berechnetes Feld in einer Abfrage



1. Die Datenbank *016Stadtlupe* ist geöffnet.
2. Legen Sie eine neue Abfrage in der SQL-Ansicht an.
3. Mit dem folgenden SELECT-Befehl lassen Sie sich einige Felder der Tabelle *Anzeigenschaltungen* anzeigen:



```
SELECT [Auftrags-Nr], Preis, Aufschlag, MwStSatz  
FROM Anzeigenschaltungen;
```

4. Jetzt wird das berechnete Feld ergänzt:

```
SELECT [Auftrags-Nr], Preis, Aufschlag, MwStSatz, ([Preis]+[Aufschlag])*(1+[MwStSatz]) AS Bruttoumsatz  
FROM Anzeigenschaltungen;
```
5. Speichern Sie die Abfrage unter dem Namen *AnzeigenUmsätze (SQL)*.

Auftrags-Nr	Preis	Aufschlag	MwStSatz	Bruttoumsatz
4	800,00 €	0,00 €	16,00%	927,99999714
5	1.200,00 €	0,00 €	16,00%	1391,9999957
4	800,00 €	0,00 €	16,00%	927,99999714
5	1.200,00 €	0,00 €	16,00%	1391,9999957
1	2.500,00 €	0,00 €	16,00%	2899,9999911
3	2.300,00 €	0,00 €	16,00%	2667,9999918
4	800,00 €	0,00 €	16,00%	927,99999714
1	2.500,00 €	0,00 €	16,00%	2899,9999911
3	2.300,00 €	0,00 €	16,00%	2667,9999918
4	800,00 €	0,00 €	16,00%	927,99999714

Datensatz: 1 von 25

Das Ergebnis der Abfrage

Ein sehr wichtiges SQL-Prinzip ist die Berechnung von Ergebnissen über Datensatzmengen mittels so genannter Aggregatfunktionen. Diese Funktionen fassen Feldinhalte ausgewählter Datensätze zusammen.

Aggregatfunktion:	Beschreibung:
SUM	Berechnet die Summe der Werte in einem Feld.
MIN	Berechnet den kleinsten Wert in einem Feld.
MAX	Berechnet den größten Wert in einem Feld.
AVG	Berechnet den Durchschnitt der Feldwerte.

Tabelle 6:
SQL Aggregatfunktionen

Diese Aggregatfunktionen setzen Sie in der nächsten Abfrage ein.

Daten mit Aggregatfunktionen zusammenfassen



Beispiel 17

- Die Datenbank *017Stadtlupe* ist geöffnet.
- Legen Sie eine neue Auswahlabfrage an.
- Wechseln Sie in die SQL-Ansicht.
- Geben Sie in das SQL-Fenster den Anfang eines SQL-Kommandos ein:
SELECT Min(Preis) AS MinPreis,
Damit legen Sie fest, dass im ersten Ergebnisfeld der kleinste Preis angezeigt werden soll. Außerdem legen Sie über das Schlüsselwort *AS* den Namen der angezeigten Spaltenüberschrift mit *MinPreis* fest.



2 DATEN MIT SQL ABFRAGEN UND ÄNDERN

5. Geben Sie nun den Rest des Kommandos ein:

Max(Preis) AS MaxPreis, Avg(Preis) AS Durchschnitt FROM Anzeigenschaltungen;



6. Führen Sie diese Abfrage aus.

7. Speichern Sie die Abfrage unter dem Namen *Preisauswertung (SQL)*.

Die Abfrage *Preisauswertung (SQL)* in der Datenblattansicht

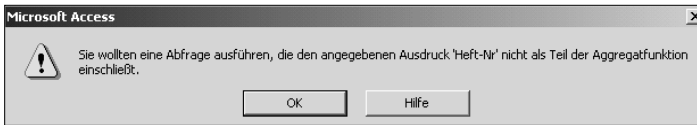
MinPreis	MaxPreis	Durchschnitt
800,00 €	5.000,00 €	1.276,00 €

In vielen Situationen möchten Sie keine Aggregatwerte über eine gesamte Tabelle berechnen lassen, sondern benötigen Zusammenfassungen nach bestimmten Gruppenfeldern. Diese Aufgabenstellung wird in SQL über das Schlüsselwort *GROUP BY* gelöst. *GROUP BY* steht immer zwischen der *FROM*-Klausel und dem *ORDER BY* (falls vorhanden). Dahinter geben Sie das Feld oder die Felder an, nach denen gruppiert werden soll. Möchten Sie beispielsweise Adressen nach Ort und Strasse zusammenfassen, so lautet die Klausel: *GROUP BY Ort, Strasse*.

Das folgende Beispiel zeigt, wie Sie jedem Heft den dazu gehörenden Anzeigenumsatz zuordnen.



- Die Datenbank *018Stadtlupe* ist geöffnet.
- Legen Sie eine neue Auswahlabfrage an.
- Wechseln Sie in die SQL-Ansicht.
- Geben Sie den ersten Teil der Abfrage ein:
`SELECT [Heft-Nr], Sum(Preis+Aufschlag) AS Auftragssumme
FROM Anzeigenschaltungen;`
- Führen Sie diese Abfrage aus. Sie erhalten eine Fehlermeldung, da Sie noch nicht angegeben haben, was mit dem Feld *Heft-Nr* passieren soll.



- Ergänzen Sie die Abfrage um den Text:
`GROUP BY [Heft-Nr];`
- Starten Sie die Abfrage
`SELECT [Heft-Nr], Sum(Preis+Aufschlag)
AS Auftragssumme
FROM Anzeigenschaltungen GROUP BY
[Heft-Nr];`

erneut. In der ersten Ergebnisspalte werden die Heft-Nummer und in der zweiten Spalte der dazu gehörende Anzeigenumsatz dargestellt.
- Speichern Sie die Abfrage unter *Auftragssumme nach Heft-Nr gruppiert (SQL)*.

Heft-Nr	Auftragssumme
NO9821	2.000,00 €
NO9822	2.000,00 €
NO9823	5.600,00 €
NO9824	5.600,00 €
NO9825	1.600,00 €
NO9826	800,00 €
NO9901	800,00 €
NO9902	800,00 €
NO9903	800,00 €
NO9904	800,00 €
NO9905	800,00 €
NO9906	800,00 €
NO9907	800,00 €
NO9908	800,00 €
NO9909	800,00 €
NO9910	800,00 €
NO9911	800,00 €
NO9912	800,00 €
S00203	5.000,00 €

Datensatz: 1 von 19

Die Abfrage *Auftragssumme nach Heft-Nr gruppiert (SQL)*

Beziehungen zwischen Tabellen werden in SQL mittels des JOIN-Befehls dargestellt. JOIN-Operationen können in der FROM-Klausel eines SQL-Befehls eingesetzt werden, um Tabellen miteinander zu verbinden. Die folgende Tabelle gibt über die Varianten Auskunft:

Tabelle 7:
Die JOIN-Klausel

JOIN-Klausel:	Beschreibung:
INNER JOIN	Zwei Tabellen werden anhand identischer Feldinhalte miteinander verknüpft.
LEFT JOIN	Es werden sämtliche Datensätze der Tabelle angezeigt, die auf der linken Seite des JOIN-Operators stehen. Dies entspricht einer Access-Inklusionsabfrage.
RIGHT JOIN	Entspricht dem LEFT JOIN, nur dass alle Datensätze der rechten Tabelle angezeigt werden.

Insgesamt stellt sich ein SQL-Befehl mit JOIN wie in dem folgenden Beispiel dar:

```
SELECT Kunden.[Kunden-Nr], Kunden.Name, Aufträge.[Auftrags-
Nr], Aufträge.Bearbeiter
FROM Kunden INNER JOIN Aufträge ON Kunden.[Kunden-Nr]
= Aufträge.[Kunden-Nr];
```

Beachten Sie, dass zuerst der Tabellenname (z. B. *Kunden*), dann der Feldname (z. B. *Name*), getrennt durch einen Punkt, eingegeben werden. Felder, die aus mehr als einem Wort bestehen, müssen in eckigen Klammern eingegeben werden.

Die Felder, durch die die beiden Tabellen verknüpft werden, werden gemeinsam mit dem Namen der Quelltabellen hinter dem Schlüsselwort ON angegeben.

Das folgende Beispiel verdeutlicht den Einsatz des JOIN-Befehls. Diese Abfrage werden wir nicht in der SQL-Ansicht erstellen.

Beispiel 19

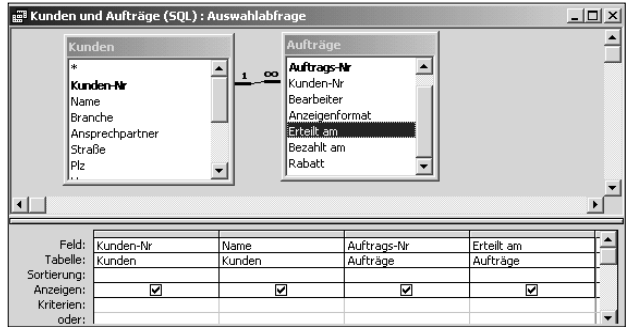


Tabellen durch INNER JOIN verknüpfen



1. Erstellen Sie in der Datenbank *019Stadtlupe* eine neue Abfrage in der Entwurfsansicht.
2. Fügen Sie der Abfrage die Tabellen *Kunden* und *Aufträge* hinzu.

- Fügen Sie dem Entwurfsbereich der Abfrage aus der Tabelle *Kunden* die Felder KUNDEN-NR und NAME hinzu sowie aus der Tabelle *Aufträge* die Felder AUFTRAGS-NR und ERTEILT AM.



- Wechseln Sie nun in die SQL-Ansicht der Abfrage.

- Betrachten Sie, wie die beiden Tabellen miteinander durch den Ausdruck INNER JOIN verknüpft werden:
SELECT Kunden.[Kunden-Nr], Kunden.Name, Aufträge.[Auftrags-Nr], Aufträge.[Erteilt am]
FROM Kunden INNER JOIN Aufträge ON Kunden.[Kunden-Nr] = Aufträge.[Kunden-Nr];

Die Abfrage wird im Abfragegenerator erstellt

- Der Ausdruck **ON Kunden.[Kunden-Nr] = Aufträge.[Kunden-Nr]** gibt an, dass die Verknüpfung über das Feld KUNDEN-NR hergestellt werden soll, welches in beiden Tabellen enthalten ist. Beachten Sie, dass die Tabellennamen mit angegeben werden.
- Überprüfen Sie das Abfrageergebnis in der Datenblattansicht. Beachten Sie, dass ausschließlich Datensätze für Kunden angezeigt werden, die tatsächlich einen Auftrag erteilt haben.

Kunden-Nr	Name	Auftrags-Nr	Ertelt am
31	Designhaus	1	02.02.2002
22	Alexis Moden	2	03.02.2002
42	Paris-Moden	3	02.02.2002
35	Intercom Design	4	04.02.2002
7	Hörning Metall-Center	5	02.02.2001
31	Designhaus	6	08.02.2002
1	AB-Fotoshop	7	03.02.2002
35	Intercom Design	8	15.02.2002
31	Designhaus	9	09.07.2002
*(AutoWert)		(AutoWert)	

Datensatz: 1 von 9

Das Abfrageergebnis in der Datenblattansicht

- Speichern Sie die Abfrage unter dem Namen *Kunden und Aufträge (SQL)*.

Sie werden im nächsten Beispiel die gerade angelegte Abfrage so modifizieren, dass auf jeden Fall sämtliche Kundendatensätze ausgegeben werden, egal ob es Aufträge gibt oder nicht. Diese Technik ist vor allem bei Auswertungen und Berichten sinnvoll, bei denen es ja oft keinen Sinn macht z. B. nur die Kunden anzuzeigen, zu denen Aufträge existieren.

Beispiel 20



LEFT JOIN in einer Abfrage verwenden

1. Die Datenbank *020Stadtlupe* ist geöffnet.
2. Öffnen Sie die gerade angelegte Auswahlabfrage *Kunden und Aufträge (SQL)* in der SQL-Ansicht.
3. Sie machen nun aus der Abfrage eine Inklusionsabfrage: Ändern Sie den Ausdruck INNER JOIN in LEFT JOIN:
`SELECT Kunden.[Kunden-Nr], Kunden.Name, Aufträge.[Auftrags-Nr], Aufträge.[Erteilt am]`
`FROM Kunden LEFT JOIN Aufträge ON Kunden.[Kunden-Nr] = Aufträge.[Kunden-Nr];`
4. Der Ausdruck gibt an, dass die Kundentabelle mit der Auftrags-tabelle verknüpft werden soll und dabei alle Kundendatensätze angezeigt werden, egal ob zugehörige Aufträge existieren oder nicht.
5. Überprüfen Sie das Verhalten der Abfrage in der Datenblattansicht. Sie erhalten wesentlich mehr Datensätze als zuvor. Es werden alle Kunden angezeigt. Von Kunden, zu denen kein Auftrag existiert, werden nur die Felder KUNDEN-NR und NAME angezeigt. Die restlichen Felder bleiben leer.
6. Speichern Sie die geänderte Abfrage unter dem neuen Namen *alle Kunden und Aufträge (SQL)*:

Hinweis →

Die Abfrage *alle Kunden und Aufträge (SQL)*

Kunden-Nr	Name	Auftrags-Nr	Erteilt am
1	AB-Fotoshop	7	03.02.2002
2	Bährle Kunststoffe		
3	Damaschke		
4	Eflerth & Co.		
5	Eskmann		
6	Gerges & Sohn		
7	Hörning Metall-Center	5	02.02.2001
8	Hugendubel		
9	Pappwerke Jansen & Hansen		
10	Klausenberg GmbH		
11	Holz Köhler		
12	Marschnik Video		
13	Mielke		
14	O.S. Luckat		
15	Paschotkas Landmetzgerei		

Datensatz: 1 von 60

Kunden und Aufträge (SQL) ab.

Sobald Sie in einer Abfrage die GROUP BY - Variante einsetzen, kann sich die Frage stellen, wie Sie bestimmte Ergebnis-Gruppen auswählen können. Mit einer WHERE -Klausel funktioniert das nicht, denn diese filtert die Daten, bevor sie zu Gruppen zusammengefasst werden. Eine Aufgabenstellung in dieser Art könnte lauten: „Fasse die Umsatzzahlen der einzelnen Kunden zu einem Gesamtumsatz zusammen. Wähle diejenigen Kunden aus, deren Gesamtumsatz über 10.000 € liegt.“

Für die Lösung solch einer Aufgabenstellung stellt SQL die HAVING - Klausel zur Verfügung. Diese ermöglicht die Anwendung eines Kriteriums für eine Gruppe. Die HAVING - Klausel steht immer hinter dem GROUP BY - Befehl und vor dem ORDER BY. Auf das HAVING folgt ein Ausdruck, der einer WHERE - Klausel ähnlich ist. Der Vergleichswert basiert allerdings in der Regel auf einem aggregierten Ausdruck. Das folgende Beispiel zeigt die Anwendung.

Gruppierte Daten mit HAVING auswählen



Beispiel 21

- Öffnen Sie in der Datenbank *021 Stadtlupe* die Abfrage *Auftragssumme nach Heft-Nr gruppiert (SQL)* in der SQL-Ansicht.
- Damit nur Hefte angezeigt werden, deren Anzeigenvolumen über 2000 € liegt, ist eine HAVING-Klausel zu ergänzen:

```
SELECT [Heft-Nr], Sum(Preis+Aufschlag) AS Auftragssumme
FROM Anzeigenschaltungen
GROUP BY [Heft-Nr]
HAVING Sum(Preis+Aufschlag) > 2000;
```
- Speichern Sie die geänderte Abfrage unter dem Namen *Auftragssumme über 2000 (SQL)*.

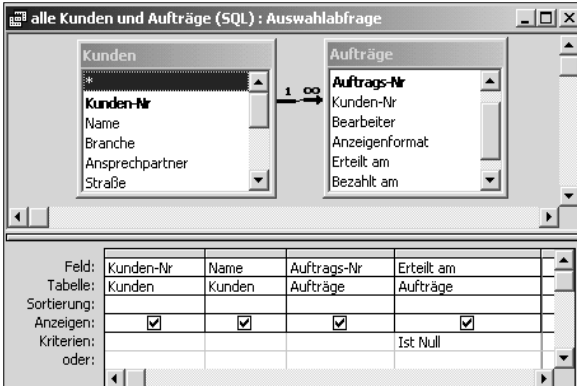
Das Ergebnis der Abfrage

Heft-Nr	Auftragssumme
NO9823	5.600,00 €
NO9824	5.600,00 €
SO0203	5.000,00 €

Datensatz: 1 von 3

Es gibt recht häufig den Wunsch, gerade diejenigen Datensätze anzuzeigen, zu denen keine Einträge in einer verknüpften Tabelle vorhanden sind. Durch eine kleine Modifikation unserer Abfrage kann dieses Ziel leicht erreicht werden.

Beispiel 22 Datensätze ohne Zuordnung finden



1. Die Datenbank *022Stadtlupe* ist geöffnet.
2. Öffnen Sie erneut die Abfrage *alle Kunden und Aufträge (SQL)* in der Entwurfsansicht.
3. Geben Sie als Kriterium für das Feld ERTEILT AM den Text *Ist Null* ein.
4. Betrachten Sie den SQL-Ausdruck:

```
SELECT Kunden.[Kunden-Nr], Kunden.Name,
Aufträge.[Auftrags-Nr], Aufträge.[Erteilt am]
FROM Kunden LEFT JOIN Aufträge ON Kunden.[Kunden-Nr] =
Aufträge.[Kunden-Nr]
WHERE (((Aufträge.[Erteilt am]) Is Null));
```

Die Abfrage *Kunden ohne Aufträge (SQL)*

5. In der WHERE-Klausel finden Sie das SQL-Kriterium *Aufträge.[Erteilt am] Is Null*. Der Ausdruck *Ist Null* im Access Abfrageoperator wird also direkt in den SQL-Ausdruck *Is Null* übersetzt.
6. Betrachten Sie das Ergebnis der Abfrage in der Datenblattansicht. Sie sehen jetzt ausschließlich Kunden, zu denen es keinen Auftrag in der Datenbank gibt. Speichern Sie die Abfrage unter dem neuen Namen *Kunden ohne Aufträge (SQL)* ab.



2.3 Fortgeschrittene Abfragetechniken mit SQL

In diesem Abschnitt geht es um die „hohe Kunst“ der SQL-Abfragen. Sie werden Aufgabenstellungen lösen, die mit den üblichen Auswahl- oder Aktualisierungsabfragen nur schwer oder gar nicht lösbar sind. So erhalten Sie in den folgenden Beispielen eine kleine Sammlung von SQL-Lösungen für Standardprobleme, die Sie leicht auf eigene Anforderungen übertragen können.

Das erste Beispiel zeigt, wie Datenmengen, die aus unterschiedlichen Tabellen stammen, mittels des SQL-Kommandos UNION verbunden werden können. Ohne diesen Befehl müssten Daten, die aus mehreren Quellen stammen, zunächst über Tabellenerstellungs- und Anfügeabfragen in einer temporären Tabelle zusammengefasst werden, bevor sie angezeigt werden können. Um daraus einen Arbeitsgang für den Anwender zu machen, müsste zusätzlich ein Makro eingesetzt werden. Dieses zu erstellen ist aufwendiger als die Verwendung einer UNION-Abfrage; zudem ist es in der Regel langsamer in der Ausführung. Sie werden nun eine Abfrage erstellen, die Adressinformationen aus zwei unterschiedlichen Tabellen in einem gemeinsamen Ergebnis sortiert zur Verfügung stellt. Dies könnte zum Beispiel die Grundlage eines Serienbriefes an alle Kunden und Mitarbeiter der Stadtlupe sein.

Abfrageergebnisse über UNION verbinden



Beispiel 23

1. Legen Sie in der Datenbank *023Stadtlupe* eine neue Auswahlabfrage an.
2. Wechseln Sie in die SQL-Ansicht.
3. Geben Sie den folgenden SQL-Befehl in das SQL-Fenster ein:

```
SELECT Name, Ansprechpartner, Straße, Plz, Ort  
FROM Kunden  
UNION  
SELECT Nachname, Vorname, Straße, Plz, Ort  
FROM Mitarbeiter  
ORDER BY Name;
```
4. Wechseln Sie in die Datenblattansicht. Kunden- und Mitarbeiterinformationen werden gemeinsam im Ergebnis dargestellt. Da eine Sortierung nach dem ersten Feld durchgeführt wurde, wechseln sich Kunden- und Mitarbeiterdatensätze ab.

2 DATEN MIT SQL ABFRAGEN UND ÄNDERN

Hinweis →

- Die Spaltennamen für das Resultat werden von der ersten Abfrage genommen. D. h. in unserem Beispiel, dass die Vornamen der zweiten Tabelle jetzt in der Spalte *Ansprechpartner* der ersten Tabelle stehen. Wenn Sie den Spaltennamen der zweiten Tabelle beibehalten möchten, dann muss eine Referenz erstellt werden. Die Abfrage sähe dann z. B. so aus:

```
SELECT Name, Ansprechpartner AS Vorname, Straße, Plz, Ort  
FROM Kunden
```

```
UNION
```

```
SELECT Nachname, Vorname, Straße, Plz, Ort
```

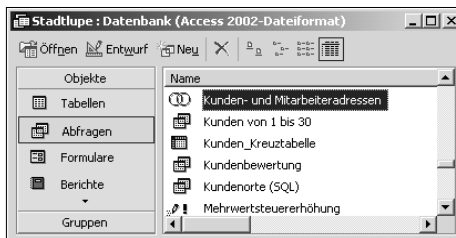
```
FROM Mitarbeiter
```

```
ORDER BY Name;
```

Hinweis →

- Und: Jede SELECT-Anweisung muss die gleiche Anzahl von Feldern zurückgeben, und zwar in identischer Reihenfolge. Die Datentypen der zusammengehörenden Felder müssen kompatibel sein.

Name	Ansprechpartner	Straße	Plz	Ort
AB-Fotshop	Schmittke	Tiedenweg 2	26169	Friesoythe
Adam	Ralf	Eisenbahnstr. 77	10997	Berlin
Aenter	Klaus	Perleberger Str. 118	10559	Berlin
Albers	Tina	Pezelsberger Straße 67	10645	Berlin
Alexis Moden		Alt-Reinickendorf 23	13407	Berlin
Anna Schmitz		Märker Grund 23	44287	Dortmund
Atelier Steinberger		Küchergartenallee 127	07548	Gera
Bährle Kunststoffe	Birkhölzer	Spandauer Str. 86	13591	Berlin
Bella Donna		Am Vogelsang 1	73108	Gammelschauser
Berliner Moden		Schwarzkrankenstraße 6	10116	Berlin



- Speichern Sie die Abfrage unter dem Namen *Kunden- und Mitarbeiteradressen* und schließen Sie diese.
- Beachten Sie, dass im Datenbankfenster diese UNION-Abfrage durch ein spezielles Symbol gekennzeichnet wird.
- Im Abfrageentwurf gibt es einen Menüeintrag *ABFRAGE/SQL SPEZIFISCH/UNION*. Man könnte hier weitere Funktionalitäten oder Hilfen vermuten. Dies ist tatsächlich nicht der Fall. Sie erhalten lediglich ein Fenster, in das Sie den SQL-Befehl eingeben können.

Hinweis →

Neben der UNION-Abfrage gibt es eine weitere, hoch interessante Abfrage-technik, die mit SQL-Kenntnissen eingesetzt werden kann. Es handelt sich dabei um so genannte Unterabfragen. Unterabfragen werden immer über ein SELECT gebildet und sehen beispielsweise so aus:

(SELECT SUM(Preis) FROM Anzeigenschaltungen).

Die runden Klammern am Anfang und am Ende der Unterabfrage sind dabei unbedingt erforderlich. Die Unterabfrage kann, je nachdem, welches Ergebnis gewünscht wird, genau einen Wert oder auch eine Liste von Felddinhalten ausgeben. *(SELECT [Kunden-Nr] FROM Kunden)* würde eine Liste sämtlicher Kundennummern zurückgeben. Die folgenden Beispiele stellen beide Varianten vor.

Eine einfache SQL-Unterabfrage einsetzen



Beispiel 24

1. Die Datenbank *024Stadtlupe* ist geöffnet.
2. Erstellen Sie eine neue Abfrage in der Entwurfsansicht.
3. Fügen Sie der Abfrage die Tabelle *Kunden* hinzu und ziehen Sie die Felder *KUNDEN-NR* und *NAME* in den Entwurfsbereich.
4. Geben Sie in die Zeile *KRITERIEN* unterhalb des Feldes *Kunden-Nr* die Bedingung:

In (SELECT [Kunden-Nr] FROM Aufträge)
ein.



Kunden-Nr	Name
1	AB-Fotoshop
2	Bährle Kunststoffe
7	Hörning Metall-Center
22	Alexis Moden
31	Designhaus
35	Intercom Design
42	Paris-Moden

Das Ergebnis der Abfrage in der Datenblattansicht

Die Auswahlabfrage mit Unterabfrage

5. Führen Sie diese Abfrage aus.
6. Es werden nur die Kunden angezeigt, die Aufträge erteilt haben.
7. Speichern Sie die Abfrage unter dem Namen *Kunden mit Aufträgen (SQL)*.

Vermutlich ist Ihnen sofort klar geworden, dass Sie diese Abfrage auch mit einer Verknüpfung der beiden Tabellen hätten gestalten können. Beim nächsten Beispiel allerdings wäre dies nicht mehr so einfach möglich.

Mit der nächsten Abfrage suchen Sie alle Einzelaufträge, deren Umsatz den allgemeinen Mittelwert übersteigt.

Beispiel 25



Eine komplexe WHERE-Klausel mit einer Unterabfrage

1. Die Datenbank *025Stadtlupe* ist geöffnet.
2. Legen Sie eine neue Auswahlabfrage an.
3. Wechseln Sie in die SQL-Ansicht.
4. Geben Sie zunächst den ersten Teil des SQL-Kommandos ein:

```
SELECT [Auftrags-Nr], Preis
FROM Anzeigenschaltungen
```

5. Es folgt die Auswahlbedingung:
`WHERE Preis > (SELECT AVG(Preis) FROM Anzeigenschaltungen);`

In der Unterabfrage berechnen Sie den Durchschnittspreis der Anzeigen. Das Ergebnis verwenden Sie in der WHERE-Klausel, um Anzeigen zu selektieren, deren Preis über diesem Wert lag.

6. Speichern Sie die Abfrage unter *Unterabfrage Durchschnittswerte (SQL)*.

	Auftrags-Nr	Preis
▶	1	2.500,00 €
	3	2.300,00 €
	1	2.500,00 €
	3	2.300,00 €
	8	5.000,00 €
*	0	0,00 €

Datensatz: 1

Das Ergebnis der Abfrage

Sie lernen jetzt ein weiteres typisches Beispiel für die Einsatzmöglichkeiten von Unterabfragen kennen. Sie zeigen dabei Kundendatensätze derjenigen Kunden an, mit denen die höchsten Umsätze erzielt wurden. Dies ist eine Aufgabenstellung, die aus zwei Schritten, der Berechnung des maximal erzielten Kundenumsatzes und der Auswahl der dazu gehörenden Kundendatensätze, besteht. Mittels Unterabfrage wird das Problem „in einem Rutsch“ gelöst. Sie lernen an dieser Stelle außerdem den Abfrageoperator *All* kennen. Er wird vor allem im Zusammenhang mit Sub-Selects benötigt. Über diesen können Sie feststellen, ob ein Vergleichswert größer (oder kleiner) als alle in der Unterabfrage vorkommenden Werte ist.

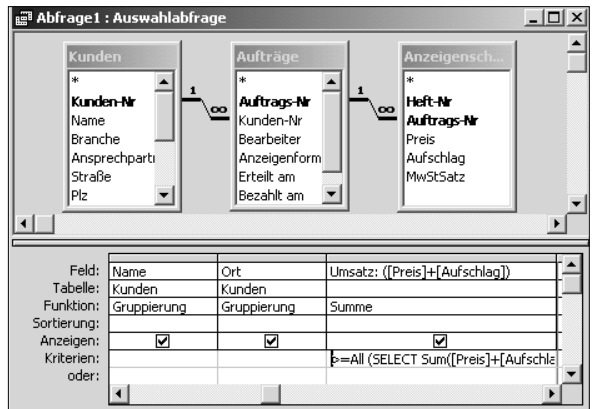


1. Die Datenbank *026Stadtlupe* ist geöffnet.
2. Erstellen Sie eine neue Abfrage in der Entwurfsansicht.
3. Fügen Sie der Abfrage die Tabellen *Kunden*, *Aufträge* und *Anzeigenschaltungen* hinzu und wählen Sie aus der Tabelle *Kunden* die Felder *KUNDEN-NR*, *NAME*, *ORT* zur Anzeige aus. Lassen Sie das Ergebnis nach der *Kunden-Nr* aufsteigend sortieren.

4. Ergänzen Sie die Abfrage um das berechnete Feld *UMSATZ*. Der Ausdruck lautet *Umsatz: ([Preis] + [Aufschlag])*.

5. Aktivieren Sie über *ANSICHT/FUNKTIONEN* die Berechnung von Aggregatwerten. Lassen Sie für das Feld *UMSATZ* die Summe berechnen.

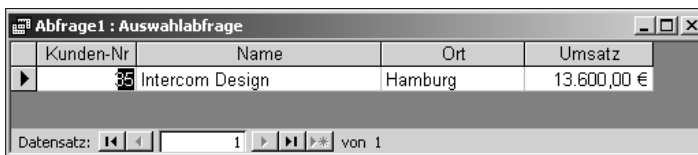
6. Sie fügen nun für das Feld *UMSATZ* eine Bedingung ein, die auf einer Unterabfrage basiert. Setzen die dazu den



Cursor in der Spalte *UMSATZ* in die Zeile *KRITERIEN*. Geben Sie dort folgenden Abfrageausdruck ein:

```
>=All (SELECT Sum([Preis]+[Aufschlag]) FROM (Aufträge INNER JOIN Anzeigenschaltungen ON Aufträge. [Auftrags-Nr] = Anzeigenschaltungen.[Auftrags-Nr]) GROUP BY [Kunden-Nr])
```

7. Führen Sie die Abfrage aus. Sie sollten jetzt nur noch denjenigen Kundensatz sehen, dessen Umsatz dem maximal erzielten Kundenumsatz entspricht.



Das Ergebnis der Abfrage

8. Speichern Sie die Abfrage unter *Unterabfrage Maximalwerte (SQL)*.

Das letzte Beispiel zum Thema „Unterabfragen“ löst ein weiteres typisches Datenbankproblem. Es geht darum, gelöschte Autowerte zu finden, um wieder eine fortlaufende Nummerierung herzustellen. In dieser Abfrage wird der Operator EXISTS eingesetzt. Über diesen lässt sich feststellen, ob ein Sub-Select überhaupt ein Ergebnis besitzt. Sie sollten sich ein wenig Zeit nehmen, um das Beispiel in Ruhe nachzuvollziehen.

Beispiel 27



Mit einer Unterabfrage fehlende Einträge suchen

1. Erstellen Sie in der Datenbank *027Stadtlupe* eine neue Abfrage in der Entwurfsansicht und fügen Sie folgenden SQL-Ausdruck hinzu:
`SELECT [Kunden-Nr] FROM Kunden K1`
2. Ergänzen Sie den Where-Teil:
`WHERE NOT EXISTS`
3. Und nun die Unterabfrage:
`(SELECT [Kunden-Nr] FROM Kunden K2 WHERE K1.[Kunden-Nr] = K2.[Kunden-Nr] - 1);`
4. Speichern Sie die Abfrage unter *Unterabfrage fehlende Einträge (SQL)*.



In der Unterabfrage wird festgestellt, ob eine Kundennummer einen Nachfolger besitzt. Dies wird über den Vergleich der Tabelle mit sich selbst in der WHERE - Klausel erreicht.

Der Ausdruck WHERE NOT EXISTS dient dazu, den Datensatz herauszufiltern, auf den die Bedingung der Unterabfrage *nicht* zutrifft. Es wird also der Datensatz mit der Kunden-Nr. angezeigt, die keinen Nachfolger besitzt.

2.4 Aktualisierungsabfragen mit SQL

Datenaktualisierungen sind in SQL über die Kommandos INSERT, UPDATE und DELETE möglich. In Access-Datenbanken, in denen keine VBA-Programmierung eingesetzt wird, spielen diese Befehle allerdings kaum eine Rolle. Wir werden hier in vier Beispielen die Basisvarianten vorstellen.

Mit einer INSERT-Abfrage Daten einfügen



Beispiel 28

1. Die Datenbank *028Stadtlupe* ist geöffnet.
2. Legen Sie eine neue Auswahlabfrage an.
3. Wechseln Sie in die SQL-Ansicht.
4. Geben Sie den ersten Teil der Abfrage ein:
INSERT INTO Kunden
Über INSERT werden in SQL neue Datensätze angelegt.
5. Der zweite Teil der Abfrage besteht aus der Feldliste
(*Name, Branche, Straße, Plz, Ort, Telefon, Telefax, eMail*)
Sie legen mit dieser Liste fest, welche Felder der Tabelle mit Werten gefüllt werden sollen.
6. Es folgt der letzte Teil:
VALUES ('Schulze' 'Mode' 'Berliner Straße' '14711' 'Berlin' '(030) 12345' '(030) 123456' 'info@schulzemode.de');
7. Gestartet wird diese Abfrage über das Symbol AUSFÜHREN.
8. Sie werden von Access darauf hingewiesen, dass Sie die Änderung nicht mehr rückgängig machen können. Klicken Sie auf JA. Sie möchten den neuen Datensatz hinzufügen.
9. Speichern Sie die Abfrage unter dem Namen *Neuer Kunde* ab und betrachten Sie das Ergebnis in der Tabelle *Kunden*.



In der Tabelle *Kunden* wurde ein neuer Kunde hinzugefügt.

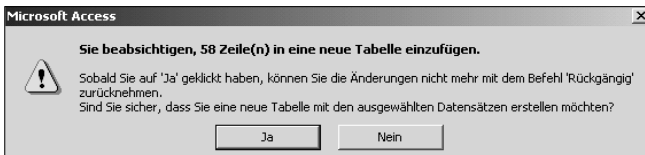
Kunden; Tabelle								
Kunden-Nr	Name	Branche	Ansprechpartner	Straße	Plz	Ort	Telefon	Telefax
50	X-Männermoden	Mode		Gradenstraße	12347	Berlin	(030) 701 80-0	(030) 703 30 19
51	Müller	Detektei	Müller	Sonnenalle	12045	Berlin	(030) 695 67 89	(030) 695 67 90
52	Müller-Lüdenscheld	Anlageberatunç	Dr. Müller	Mendenerweg	13099	Berlin		
53	Müller-Lenzig	Gesundheit		Schloßstraße	14000	Berlin		
54	Meyer & Partner	Detektei	Meyer	Kurfürstendamm	17099	Berlin		
55	Meier	Anlageberatunç	Meier	Horstweg	15000	Berlin		
56	Meyer	Foto	Meyer	Uhlandstraße	16000	Berlin		
57	Beta Design	Wohnen		Enich-Baron-Weg	12623	Berlin	(030) 527 66 -0	(030) 527 32 12
59	Schulze	Mode		Berliner Straße	14711	Berlin	(030) 12345	(030) 123456
*	(AutoWert)							

Beispiel 29



Mit SELECT INTO eine neue Tabelle erzeugen

- Legen Sie in der Datenbank *029Stadtlupe* eine neue Abfrage an und geben Sie in der SQL-Ansicht die Zeile:
SELECT Plz, Ort INTO Orte
ein. Damit legen Sie fest, dass eine neue Tabelle mit zwei Feldern angelegt werden soll.
- Die Inhalte der neuen Tabelle sollen aus den vorhandenen Kundendaten generiert werden. Der nächste Teil des SQL-Befehls lautet daher:
FROM Kunden;
- Führen Sie die Abfrage aus.



Plz	Ort
30169	Friesoythe
13591	Berlin
12347	Berlin
07545	Gera
13407	Berlin
12456	Berlin
13599	Berlin
12161	Berlin
26169	Friesoythe
10437	Berlin
06944	Dessau
10318	Berlin

Die neue Tabelle Orte

- Die Frage, ob Sie sicher sind, dass Sie eine neue Tabelle mit den ausgewählten Datensätzen erstellen möchten, beantworten Sie mit JA.
- Anschließend sollte sich eine neue Tabelle (*Orte*) in Ihrer Datenbank befinden.
- Speichern Sie die Abfrage unter *Ortsliste erstellen* ab.

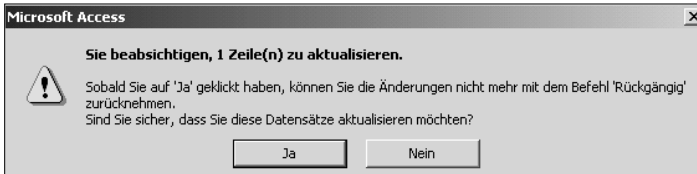
Beispiel 30



Mit UPDATE Abfrage-Daten ändern

- Die Datenbank *030Stadtlupe* ist geöffnet.
- Geben Sie in der SQL-Ansicht einer neuen, leeren Abfrage als ersten Teil eines SQL-Befehls die Zeile:
UPDATE Kunden
ein. Sie legen damit fest, dass eine Änderung der Kundendaten vorgenommen werden soll.
- Der zweite Teil der Abfrage lautet:
SET Straße = ‚Kurfürstendamm 119‘
Dies legt die Änderung an den ausgewählten Datensätzen fest.
Führen Sie die Abfrage auf gar keinen Fall jetzt schon aus!

- Über die WHERE Klausel:
 $WHERE [Kunden-Nr] = (SELECT Max([Kunden-Nr]) FROM Kunden);$
 bestimmen Sie noch, dass die Änderung im Datensatz mit der höchsten Kunden-Nr ausgeführt werden soll. Das ist der Datensatz, den Sie im vorletzten Beispiel eingefügt hatten.
- Testen Sie die Abfrage über AUSFÜHREN, klicken Sie wieder auf JA und speichern Sie diese unter dem Namen *Kunden ändern* ab.



Ein Datensatz in der Tabelle Kunden wurde geändert.

Kunden-Nr	Name	Branche	Ansprechpartner	Straße	Plz	Ort	Telefon
50	X-Männermoden	Mode		Gradestraße	12347	Berlin	(030) 701 80-0
51	Müller	Detektei	Müller	Sonnenalle	12045	Berlin	(030) 695 67 89
52	Müller-Lüdenschaid	Anlageberatung	Dr. Müller	Mendenerweg	13099	Berlin	
53	Müller-Lenzig	Gesundheit		Schloßstraße	14000	Berlin	
54	Meyer & Partner	Detektei	Meyer	Kurfürstendamm	17099	Berlin	
55	Meier	Anlageberatung	Meier	Horstweg	15000	Berlin	
56	Meyer	Foto	Meyer	Umlandstraße	16000	Berlin	
57	Beta Design	Wohnen		Erich-Baron-Weg	12623	Berlin	(030) 527 66-0
59	Schulze	Mode		Kurfürstendamm 119	14711	Berlin	(030) 12345

Datensatz: (AutoWert) von 58

Mit einer DELETE-Abfrage Daten löschen



Beispiel 31

- Erstellen Sie in der Datenbank *031Stadtlupe* eine neue Abfrage und wechseln Sie in die SQL-Ansicht.
- Die erste Zeile des einzugebenden Befehls lautet:
 $DELETE FROM Kunden$
- Sollten Sie diese Abfrage jetzt ausführen, würden Sie alle Informationen in der Tabelle *Kunden* löschen!
- Geben Sie in die zweite Zeile den Abfrageausdruck:
 $WHERE [Kunden-Nr] = (SELECT Max([Kunden-Nr]) FROM Kunden);$
 ein. Sie legen damit fest, dass der zuletzt eingegebene Datensatz entfernt werden soll.
- Führen Sie die Löschung durch und speichern Sie die Abfrage unter *Kunden löschen* ab.
- Überprüfen Sie in der Tabelle *Kunden*, ob der Datensatz mit der Kunden-Nr. 59 gelöscht wurde.

