



Abfragen mit SQL (Teil 1)

Daten-Abfragen (englisch: queries) werden mit der Anweisung **SELECT** ausgeführt. Der Befehl **SELECT** liefert eine Ergebnistabelle. Der **SELECT**-Befehl ist sehr mächtig und wird noch in aller Ausführlichkeit besprochen. Zunächst einige einfache Varianten:

Abfragen aller Tupel mit allen Attributen

```
SELECT * FROM Tabellename
```

```
SELECT * FROM PERSON
```

Mit dem Stern * werden alle Attribute einer Relation ausgewählt. Hinter **FROM** steht der Name der anzusprechenden Tabelle.

Abfragen aller Tupel mit einem oder einigen Attributen

```
SELECT F1 FROM Tabelle
```

```
SELECT F1[, F2 [, ...]] FROM Tabelle
```

```
SELECT NAME FROM PERSON
```

```
SELECT VORNAME, NAME FROM PERSON
```

Es wird eine Liste der auszugebenden Spalte angegeben; die Spaltennamen werden durch Kommata voneinander getrennt. Die Attribute werden in der angegebenen Reihenfolge gelistet.

Abfragen von Tupel mit Auswahlkriterien

```
SELECT F1[, F2 [, ...]] FROM Tabelle
WHERE Feld Operator Wert
```

```
SELECT NAME FROM PERSON
WHERE PLZ = 53024
```

Mit der Klausel **WHERE** wird ein Kriterium formuliert. Es können nicht nur numerische Werte abgefragt werden. Auch Zeichenketten können als Kriterium gelten. Diese sind dann in Hochkommata zu setzen (z.B. **Name = 'Homm'**). Teilstrings können mit dem **Like**-Operator verwendet werden (z.B. **Name Like 'Ho*'** listet alle Tupel, bei denen Name mit "Ho" beginnt.)

Vergleichsoperatoren

>	Größer als	<=	Kleiner gleich
<	Kleiner als	<>	Ungleich
>=	Größer gleich	Like	Vergleich von Strings und Teilstrings

Sortierte Ausgabe der Tupel

```
SELECT F1[, F2 [, ...]] FROM Tabelle
WHERE Feld Operator Wert
ORDER BY Feld1[, Feld2...] [DESC]
```

```
SELECT NAME FROM PERSON
WHERE PLZ = 53024
ORDER BY NAME
```

Mit dem Zusatz **ORDER BY** kann eine Sortierung bei der Ausgabe nach dem angegebenen Feld vorgenommen werden. Fügt man hinter dem Feldnamen oder der -liste den Zusatz **DESC** an, erreicht man eine absteigende Reihenfolge (Standard ist aufsteigend).



Ausschließen inhaltlich gleicher Tupel

In manchen Tabellen existieren Attribute, in denen in verschiedenen Tupeln bestimmte Werte mehrfach vorkommen können (z.B. in der Tabelle PERSON das Attribut LAND). Will man nun lediglich wissen, welche Werte überhaupt vorkommen und keine Wiederholungen erhalten, nutzt man das Schlüsselwort **DISTINCT**.

```
SELECT DISTINCT Feld FROM Tabelle
```

```
SELECT DISTINCT LAND FROM PERSON
```

Abfragen mit Datumswerten als Kriterium

Datumswerte müssen in Select-Abfragen in einem bestimmten Format vorliegen, das sich an der amerikanischen Darstellungsform eines Datums orientiert.

```
SELECT * FROM Tabelle  
WHERE Datumsfeld = #MM-DD-YYYY#  
oder  
WHERE Datumsfeld = #MM/DD/YYYY#  
oder  
WHERE Datumsfeld = DATEVALUE('DD.MM.YYYY')
```

Weitere nützliche Funktionen im Zusammenhang mit Datumswerten sind `Day()`, `Month()` oder `Year()`. Als Argument wird ein Attribut mit dem Datentyp `Date` übergeben. Nun kann man das Ergebnis des Funktionsaufrufes mit einem Wert vergleichen.

```
SELECT * FROM Tabelle  
WHERE DAY(Datumsfeld) = 21  
oder  
WHERE MONTH(Datumsfeld) = 1  
oder  
WHERE YEAR(Datumsfeld) = 1961
```

Abfragen von NULL-Werten

Um festzustellen, ob in einer Tabelle NULL-Werte enthalten sind (**die man vermeiden sollte!**), lässt sich folgende Abfrage für ein Attribut verwenden.

```
SELECT * FROM Tabelle WHERE Attribut IS NULL
```

Nochmaliger Hinweis: Ein NULL-Wert ist weder die Zahl 0 noch eine leere Zeichenkette. Es ist ein nicht initialisiertes Attribut eines Tupels.



B	Lehrgang: Datenbanken	Arbeitsblatt Nr. 15
S	Thema: Abfragen mit SQL (Teil 1)	Datum:
G	Name:	Seite 3 von 3

Übungen zu Abfragen mit SQL (Teil 1)

Verwenden Sie die Access Datenbank Telefon.mdb für die nachfolgenden Übungen. Verändern Sie nicht die Daten in den Tabellen. Speichern Sie jedoch die getätigten Abfragen; am besten unter der Aufgabennummer. Sehen Sie sich zu Beginn die Struktur der Tabellen in der Entwurfsansicht an, um sich einen Überblick über die Attributbezeichnungen und deren Datentypen zu verschaffen.

1. Listen Sie den gesamten Datenbestand
2. Ermitteln Sie die Namen und Vornamen sowie Geburtstage aller Personen
3. Listen Sie die Vorname und Name alphabetisch nach Name sortiert
4. Listen Sie Vorname, Name und Ort aller Personen aus Bonn
5. Listen Sie Vorname, Name und Plz aller Personen mit der Plz 53024 sortiert nach Name
6. Listen Sie Vorname, Name und Plz aller Personen mit Postleitzahlen größer gleich 50000
7. Listen Sie Vorname, Name und Plz aller Personen mit Postleitzahlen kleiner 50000
8. Listen Sie Name und Geburtstag aller Personen, deren Name mit einem 'D' beginnt
9. Listen Sie Vorname, Name und Ort aller Personen, deren Name mit 'A' bis 'C' beginnt
10. Listen Sie Name und Ort aller Personen, die nicht in Bonn wohnen, sortiert nach Name
11. Listen Sie alle Plz und Ortsnamen, sortiert nach Ortsname
12. Listen Sie alle Städte, aber jede nur einmal!
13. Listen Sie alle Länder in **absteigender** Reihenfolge, aber jedes nur einmal!
14. Listen Sie alle Personen, die am 17. November 1633 Geburtstag haben
15. Listen Sie alle Personen, die im Monat Mai Geburtstag haben, sortiert nach Tag
16. Listen Sie alle Personen, die im Jahr 1957 Geburtstag haben
17. Listen Sie alle Personen, die ab 1940 Geburtstag haben, sortiert nach Jahr
18. Listen Sie allen Datensätze, deren Plz-Wert NULL ist
19. Listen Sie allen Datensätze, deren Plz-Wert **eine** leere Zeichenkette ist
20. Listen Sie allen Datensätze, deren Plz-Wert **keine** leere Zeichenkette ist
21. In welchen Ergebnistabellen aus den drei vorherigen Abfragen findet sich der Datensatz 0?