

## Grundlagen

SQL ist eine standardisierte Abfragesprache, mit der sich sämtliche Arbeiten im Zusammenhang mit relationalen Datenbanken erledigen lassen.

SQL steht für Structured Query Language - zu deutsch etwa: strukturierte Abfragesprache (Aussprache: „es qu/quju el“ bzw. „sequel“). Es handelt sich um eine sogenannte nichtprozedurale Sprache (im Gegensatz zu prozeduralen Sprachen).

Nichtprozedural bedeutet »was« statt »wie«. SQL beschreibt, welche Daten abzurufen, zu löschen oder einzufügen sind und nicht, wie das zu geschehen hat.

SQL erblickte Ende der 70er Jahre in einem IBM-Labor in San Jose (Kalifornien) das Licht der Welt. Dies geschah kurz nach der Entwicklung des Konzepts für relationale Datenbanken durch Dr. E.F. Codd. Die Entwicklung war ursprünglich für das IBM-Produkt DB2 vorgesehen.

SQL wurde als eine leicht zu erlernende Abfragesprache konzipiert - zu einem Zeitpunkt, als Rechner noch keine anwenderfreundlichen Benutzeroberflächen besaßen.

SQL basiert auf der natürlichen Sprache Englisch und ist daher relativ einfach zu erlernen und handzuhaben. Sie ist nichts desto trotz ein sehr effizientes Werkzeug um Informationen aus einer Datenbank auszulesen.

SQL ist keine Programmiersprache im herkömmlichen Sinn, da sie keine eigenständigen Programmschritte unterstützt. Es handelt sich um eine reine Abfragesprache, die allein zur Manipulation von relationalen Datenbanken dient.

Im folgenden sollen vier wichtige SQL-Anweisungen vorgestellt werden, mit der sich Daten abfragen, hinzufügen, ändern und löschen lassen. Grundlage ist eine fiktive Tabelle *kunden* mit den Feldern *kundennr*, *zuname*, *vorname*, *gebdatum*, *plz*, *ort*, *umsatz* und *skonto*.

In den nachfolgenden Beispielen wird aus Gründen der Übersichtlichkeit vor jedem Schlüsselwort eine neue Zeile begonnen. Dies ist bei SQL nicht erforderlich. Man kann jede SQL-Anweisung in einer Zeile schreiben.

## Die SELECT-Anweisung

Die Anweisung für eine Anfrage an eine relationale Datenbank lautet SELECT und entspricht einer Abfrage.

Die einfachste SQL-Anweisung lautet:

```
SELECT *  
FROM kunden
```

Mit diesem Befehl werden alle Felder (\*) aus der Tabelle *kunden* angefordert.

Mit der Anweisung

```
SELECT zuname, vorname  
FROM kunden
```

werden die Felder *zuname* und *vorname* aus der Tabelle *kunden* angefordert bzw. vom Datenbankmanagementsystem geliefert.

Ein sehr wichtiger Parameter für SELECT-Anweisungen ist WHERE:

```
SELECT zuname, vorname  
FROM kunden  
WHERE plz = 3300
```

Mit dieser Anweisung werden alle Kunden (Datensätze) geliefert, deren Wert im Feld *plz* 3300 beträgt.

Schließlich kann man mit dem Parameter ORDER BY die Datensätze noch sortieren:

```
SELECT zuname, vorname  
FROM kunden  
WHERE plz = 3300  
ORDER BY zuname
```

Bei dieser Abfrage sind vier typische Schlüsselwörter des SELECT-Befehls sichtbar:

- **SELECT** Befehlswort (=Abfrage)
- **FROM** Angabe der Datenquelle
- **WHERE** Angabe der Kriterien
- **ORDER BY** Sortierung

Des Weiteren verfügt die SELECT-Anweisung noch über die Parameter:

- **GROUP BY** zum Zusammenfassen
- **HAVING** zur Angabe der Kriterien im Zusammenhang mit GROUP BY
- **JOIN** Verknüpfung von Tabellen

**WHERE (optional):** Mit WHERE können Kriterien angegeben werden. Alle Datensätze, die diesen Kriterien entsprechen werden zurückgeliefert.

WHERE kann bis zu 40 Kriterien enthalten, die mit OR bzw. AND verknüpft werden.

```
SELECT *
FROM kunden
WHERE zuname = 'MAIER'
      AND umsatz > 20000
```

Folgende Operatoren sind möglich:

= bzw. <>	gleich / ungleich
< bzw. <=	kleiner / kleiner gleich
> bzw. >=	größer / größer gleich
LIKE	wie
BETWEEN	zwischen

Zahlen (numerische Werte)	123
Text (String)	'abc'
Datum	#12/31/99#

Beispiele:

% statt \* in ANSI-SQL

```
SELECT *
WHERE gebdatum = #5/10/76#

SELECT *
FROM kunden
WHERE zuname Like 'S*'

SELECT *
FROM kunden
WHERE umsatz BETWEEN 9000 AND 20000
```

(\* bzw. % ersetzt andere Zeichen)

**ORDER BY (optional):** Mit ORDER BY werden die Datensätze entweder aufsteigend oder absteigend sortiert:

- ASC = aufsteigend
- DESC = absteigend

```
SELECT *
FROM kunden
ORDER BY zuname ASC bzw. DESC
```

Wird weder ASC noch DESC angegeben, so werden die Datensätze standardmäßig aufsteigend sortiert.

ORDER BY ist normalerweise die letzte Angabe in einer SELECT-Anweisung.

**GROUP BY (optional):** GROUP BY fasst Datensätze, die in den angegebenen Feldern

dieselben Werte enthalten, zu einem einzelnen Datensatz zusammen:

```
SELECT plz
FROM kunden
GROUP BY plz
```

Diese Anweisung gruppiert alle Datensätze aus der Tabelle *kunden* nach dem Feld *plz*.

```
SELECT plz, Count(plz) AS anzahl
FROM kunden
GROUP BY plz
ORDER BY Count(plz) DESC
```

Mit dieser Anweisung wird eine Liste mit der Anzahl der Kunden je Postleitzahl ausgegeben, die absteigend sortiert wird. Dabei wird die Funktion count() verwendet. Weitere Funktionen: max(), min(), sum(), avg().

**HAVING (optional):** HAVING kann nur im Zusammenhang mit GROUP BY verwendet werden. Nachdem GROUP BY die Datensätze gruppiert, kann man mit HAVING durch die Angabe von Kriterien bestimmte Datensätze auswählen (ähnlich wie bei WHERE):

```
SELECT plz, Count(plz) AS anzahl
FROM kunden
GROUP BY plz
HAVING count(plz) > 100
```

Man verwendet den WHERE-Parameter, um Zeilen auszuschließen, die nicht gruppiert werden sollen, und verwendet HAVING, um Datensätze nach dem Gruppieren zu filtern.

**JOIN (optional):** Mit JOIN kann man mehrere Tabellen über ein bestimmtes Feld verknüpfen um Felder aus mehreren Tabellen anzuzeigen. In diesem Fall wird vor dem Feldnamen der Tabellenname angegeben.

```
SELECT kunden.zuname, kunden.plz,
       postleitzahlen.ort
FROM kunden
INNER JOIN postleitzahlen
ON kunden.plz = postleitzahlen.plz
```

Bei diesem Beispiel werden die Tabellen *kunden* und *postleitzahlen* mit INNER JOIN über die Felder *kunden.plz* und *postleitzahlen.plz* verknüpft. INNER JOIN bedeutet, dass nur jene Datensätze angezeigt werden, bei denen

der Inhalt der verknüpften Felder identisch sind.

*Da bei mehreren Tabellen mit ev. unterschiedlichen Verknüpfungsarten die SELECT-Anweisungen sehr komplex werden, an dieser Stelle ein kleiner Tipp: Erstellen Sie mit Access die gewünschte Abfrage und wechseln Sie anschließend in die Ansicht "SQL".*

## Die INSERT-Anweisung

Mit der INSERT-Anweisung kann man Datensätze zu einer Tabelle hinzufügen:

```
INSERT
INTO kunden (Feld1, Feld2, Feld3)
VALUES(Wert1, Wert2, Wert3)
```

Mit dieser Anweisung wird ein neuer Datensatz in die Tabelle *kunden* hinzugefügt. In der ersten Klammer stehen die Felder, die gefüllt werden sollen, in der zweiten Klammer stehen die Werte (Variable oder Konstante) mit denen die Felder aus der ersten Klammer gefüllt werden.

```
INSERT
INTO kunden (zuname,vorname,umsatz)
VALUES('Meier','Heinrich',20000)
```

Achtung:

- die Reihenfolge der Werte in der zweiten Klammer müssen genau der Reihenfolge in der ersten Klammer entsprechen
- die Anzahl der Werte in der zweiten Klammer müssen genau der Anzahl in der ersten Klammer entsprechen
- die Datentypen der Variablen/Konstanten müssen genau den Datentypen der Felder entsprechen

## Die UPDATE-Anweisung

Mit der UPDATE-Anweisung kann man die Inhalte eines Datensatzes ändern.

```
UPDATE kunden
SET vorname = 'Walter', plz = 3300
WHERE kundennr = 10
```

Diese Anweisung ändert die Felder *vorname* und *plz* in der Tabelle *kunden* und zwar beim Kunden mit der *kundennr* 10.

Mit UPDATE kann man auch den Inhalt von Feldern neu berechnen:

```
UPDATE kunden
SET umsatz = umsatz + 3400
WHERE kundennr = 10
```

In diesem Fall wird beim Kunden mit der *kundennr* 10 der Inhalt des Feldes *umsatz* um 3.400 erhöht.

Parameter WHERE: siehe SELECT.

## Die DELETE-Anweisung

Zu guter Letzt gibt es natürlich auch noch eine Anweisung zum Löschen von Datensätzen:

```
DELETE
FROM kunden
WHERE kundennr = 10
```

Mit dieser Anweisung wird der Kunde mit der *kundennr* 10 aus der Tabelle *kunden* gelöscht.

Achtung: Fehlt die Angabe von WHERE, so werden allen Datensätze dieser Tabelle gelöscht.

Parameter WHERE: siehe SELECT.

## Zusammenfassung

SQL ist ein Standard zur Behandlung von relationalen Datenbanken. Wichtige Anweisungen sind:

SELECT	Auswahlabfrage
INSERT	Hinzufügen
UPDATE	Ändern
DELETE	Löschen

Während SELECT nur Werte aus einer Tabelle ausliest, werden mit INSERT, UPDATE und DELETE die Inhalte von Tabellen verändert.

MS Access unterstützt SQL, d.h., in Access können sämtlichen Tabellenoperationen mit SQL-Anweisungen durchgeführt werden.

Diese vier Anweisungen stellen nur einen kleinen, wenn auch wichtigen Teil von SQL dar.

*Beispiele für relationale DB-Management-systeme, die SQL unterstützen: Oracle, SQL-Server, MySQL, MS Access.*