Abfragen von Datensätzen mit der DQL (engl. *data query language*)

Die wichtigste Aufgabe von SQL besteht darin, Informationen aus Datenbanken zu gewinnen. Daher ist der SELECT-Befehl aus der Datenabfragesprache DQL (Teilmenge von SQL) für die meisten Datenbankanwender die bedeutsamste Anweisung.

Eine SELECT-Anweisung besteht nur aus wenigen Bausteinen, die jedoch auf vielfältige Weise kombiniert werden können. Damit diese Einführung überschaubar bleibt, werden manche Bausteine und Kombinationen erst im *Aufbaumodul 2.3.2 Datenbanksysteme II* eingehend behandelt.

Das Grundgerüst der SELECT-Anweisung

Die einzelnen Bausteine werden weiter unten noch beispielhaft vorgestellt. Damit du aber jetzt schon siehst, wohin die Reise geht, gibt es hier bereits vorab die Syntax der SELECT-Anweisung:

SELECT [DISTINCT | ALL] **Ausdrucksliste**
FROM Tabelle [Tabellenverbund mit **JOIN**]
[**WHERE Bedingungsliste**]
[**GROUP BY** Ausdrucksliste]
[**HAVING** Bedingungsliste]
[**ORDER BY Ausdrucksliste** [ASC[1] | DESC[2]]]

Erläuterungen:

- **BLAU** und **FETT**: SQL-Schlüsselwörter
- Graue Schrift: Diese Bausteine werden hier nicht oder nur am Rande behandelt.
- **Ausdrucksliste**: Liste von Datenfeldern (aus einer oder mehreren Tabellen) oder Berechnungen.
- **Bedingungsliste**: Eine oder mehrere Bedingungen mit logischen Operatoren und Vergleichsoperatoren.

Vergleichs-Operatoren:

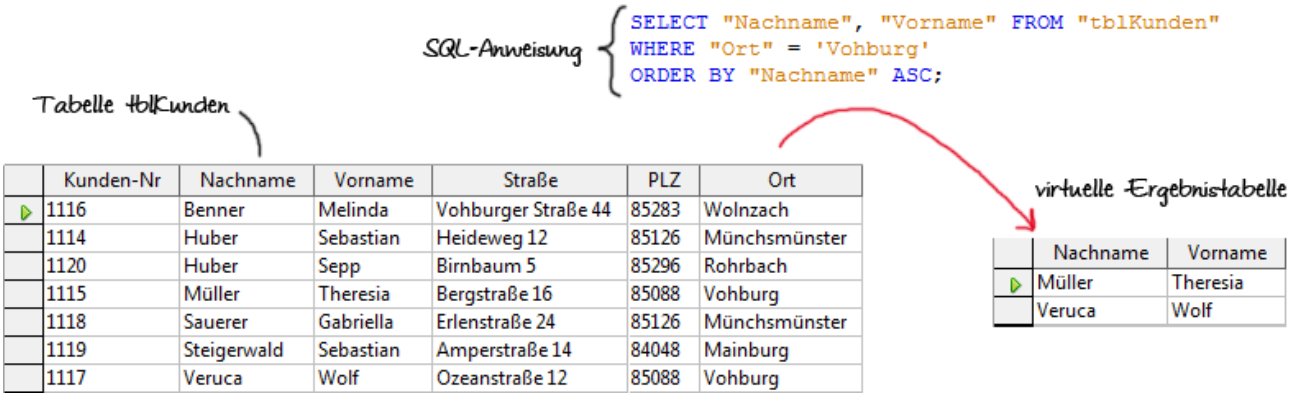
>	Größer als (geht auch mit größer-gleich, also >=)
<	Kleiner als (geht auch mit kleiner-gleich, also <=)
=	Gleich
!=	Ungleich (geht auch mit <>)
BETWEEN	Zwischen Minimal- und Maximalwert
LIKE	Zeichenkettenvergleich mit **Platzhaltern** **%** steht für eine beliebige Zeichenkette (mit 0 oder mehr Zeichen) _ steht für ein beliebiges einzelnes Zeichen
IN	Der IN-Parameter vergleicht, ob der Inhalt einer Spalte in einer angegebenen Liste enthalten ist.

Logische Operatoren:

OR	Logisches ODER (geht auch mit ||)
AND	Logisches UND (geht auch mit &&)
NOT	Logisches NICHT

[1] *engl. **asc**ending -> dt. aufsteigend*
[2] *engl. **desc**ending -> dt. absteigend*

Einführungsbeispiel:



Kunden-Nr	Nachname	Vorname	Straße	PLZ	Ort
1116	Benner	Melinda	Vohburger Straße 44	85283	Wolnzach
1114	Huber	Sebastian	Heideweg 12	85126	Münchsmünster
1120	Huber	Sepp	Birnbaum 5	85296	Rohrbach
1115	Müller	Theresia	Bergstraße 16	85088	Vohburg
1118	Sauerer	Gabriella	Erlenstraße 24	85126	Münchsmünster
1119	Steigerwald	Sebastian	Amperstraße 14	84048	Mainburg
1117	Veruca	Wolf	Ozeanstraße 12	85088	Vohburg

Nachname	Vorname
Müller	Theresia
Veruca	Wolf

Abb. 01: Beispiel für eine SQL-Anweisung mit Ausgangstabelle und Ergebnistabelle.

Der SQL-Anweisungblock aus dem oberen Einführungsbeispiel bewirkt:

- Eine Auswahl der Spalten (Datenfelder) Nachname und Vorname aus der Tabelle tblKunden.
  SELECT Nachname, Vorname FROM tblKunden.
- Eine Auswahl an Datensätzen (Zeilen) für die die Bedingung **Ort** = 'Vohburg' WAHR ist.
  WHERE Ort = 'Vohburg'
- Eine aufsteigende Sortierung der Datensätze nach den Werten in der Spalte Nachname.
  ORDER BY Nachname ASC (*engl. **asc**ending -> dt aufsteigend*)
- Das abschließende Kopieren der **Ergebnismenge** in eine **virtuelle Tabelle** im Arbeitsspeicher.

Weitere Beispiele zur SELECT-Anweisung

Für die ersten Beispiele verwenden wir eine vereinfachte Kundentabelle (s. Abb. 02) mit nur 7 Datensätzen und sechs Datenfeldern. So erkennst du die Auswirkungen der vorgestellten SQL-Anweisungen schon beim Lesen und Vergleichen der Ausgangstabelle mit der jeweiligen Ergebnistabelle.

Die SQL-Beispiele findest du auch als einfache Textdateien im Ordner *231-materialien/SQL/DQL*. Das erlaubt dir, die SQL-Anweisungen schnell und unkompliziert in der Datenbank *werkzeuge-00.odb* über Kopieren und Einfügen zu testen.

Ausgangstabelle für Beispiele 1 bis 4 (*s. werkzeuge-00.odb*)

Kunden-Nr	Nachname	Vorname	Straße	PLZ	Ort
1114	Huber	Sebastian	Heideweg 12	85126	Münchsmünster
1115	Müller	Theresia	Bergstraße 16	85088	Vohburg
1116	Benner	Melinda	Vohburger Straße 44	85283	Wolnzach
1117	Veruca	Wolf	Ozeanstraße 12	85088	Vohburg
1118	Sauerer	Gabriella	Erlenstraße 24	85126	Münchsmünster
1119	Steigerwald	Sebastian	Amperstraße 14	84048	Mainburg
1120	Huber	Sepp	Birnbaum 5	85296	Rohrbach

Abb. 02: Tabelle *tblKunden* aus *werkzeuge-00.odb*

Beispiel 1 (*select-where.txt)*

```
-- ********************************
-- Ermittelt die Kunden aus tblKunden,
-- die in Münchsmünster wohnen.
-- Mit dem Platzhalter * werden alle
-- Datenfelder angezeigt.
-- ********************************

SELECT * FROM "tblKunden"
WHERE "Ort" = 'Münchsmünster';
```

Kunden-Nr	Nachname	Vorname	Straße	PLZ	Ort
1114	Huber	Sebastian	Heideweg 12	85126	Münchsmünster
1118	Sauerer	Gabriella	Erlenstraße 24	85126	Münchsmünster

Abb. 03: Von Beispiel 1 erzeugte virtuelle Tabelle

Beispiel 2 (*select-order-by.txt)*

```
-- ********************************
-- Ermittelt alle Kunden aus tblKunden,
-- und sortiert sie aufsteigend
-- ASC (engl. ascending)
-- oder absteigend
-- DESC (engl. descending)
-- zuerst nach dem Nachnamen, und dann
-- nach dem Vornamen.
-- ********************************

SELECT "Nachname", "Vorname" FROM "tblKunden"
ORDER BY "Nachname" ASC, "Vorname" ASC;

-- ******************************
-- Also kommt Huber, Sebastian vor
-- Huber, Sepp. Ist doch klar, oder? ;-)
-- ******************************
```

Nachname	Vorname
Benner	Melinda
Huber	Sebastian
Huber	Sepp
Müller	Theresia
Sauerer	Gabriella
Steigerwald	Sebastian
Veruca	Wolf

Abb. 04: Ergebnismenge Beispiel 2

Beispiel 3 (*select-where-or.txt)*

```
-- ***********************************
-- Ermittelt die Kunden aus tblKunden,
-- die in Münchsmünster ODER Vohburg
-- wohnen - absteigend sortiert nach Ort.
-- ***********************************

SELECT "Nachname","Straße", "Ort" FROM "tblKunden"
WHERE "Ort" = 'Münchsmünster' OR "Ort" = 'Vohburg'
ORDER BY "Ort" DESC;
```

Nachname	Straße	Ort
Veruca	Ozeanstraße 12	Vohburg
Müller	Bergstraße 16	Vohburg
Sauerer	Erlenstraße 24	Münchsmünster
Huber	Heideweg 12	Münchsmünster

Abb. 05: Ergebnis Beispiel 3

Beispiel 4 (*select-where-not-like-%.txt)*

```
-- ************************************
-- Ermittelt die Kunden aus tblKunden,
-- in deren Datenfeld "Straße" nicht die
-- Zeichenkette 'straße' vorkommt.
-- ************************************

SELECT "Nachname","Straße", "Ort" FROM "tblKunden"
WHERE "Straße" NOT LIKE '%straße%';[3]
```

Nachname	Straße	Ort
Huber	Heideweg 12	Münchsmünster
Benner	Vohburger Straße 44	Wolnzach
Huber	Birnbaum 5	Rohrbach

Abb. 06: Ergebnis Beispiel 4

[3] Warum wird der Datensatz mit der 'Vohburger Straße 44' trotzdem angezeigt? Sieh dir das Begleitvideo an!

Ausgangstabelle für Beispiele 5 bis 8 (*s. werkzeuge-00.odb*)

	Artikel-Nr	Bezeichnung	Bestand	Preis	H-Nr
	101	Bohrmaschine	25	89,00	40122
	102	Bohrmaschine	27	129,95	40103
	103	Kelle	7	15,98	40307
	105	Pinsel	210	3,60	40307
	117	Schraubendreher	84	4,49	40122
	121	Hammer	150	5,95	40122
	122	Zange	35	11,49	40122

Abb. 07: Tabelle *tblArtikel* aus *werkzeuge-00.odb*

Beispiel 5 (*select-where-in.txt)*

```
-- *********************************
-- Liefert alle Datensätze aus tblArtikel,
-- mit den Bezeichnungen 'Bohrmaschine',
-- 'Hammer' und 'Zange'.
-- *********************************

SELECT "Artikel-Nr", "Bezeichnung", "H-Nr" FROM "tblArtikel"
WHERE "Bezeichnung" IN ('Bohrmaschine','Hammer','Zange');
```

Artikel-Nr	Bezeichnung	H-Nr
101	Bohrmaschine	40122
102	Bohrmaschine	40103
121	Hammer	40122
122	Zange	40122

Abb. 08: Ergebnis Beispiel 5

Beispiel 6 (*select-where-or-and-not.txt)*

```
-- ************************************************************
-- Liefert alle Datensätze aus tblArtikel, mit der Bezeichnung 'Bohrmaschine'
-- ODER 'Hammer', die NICHT vom Hersteller mit der H-Nr 40103 sind.
-- Beachte: Die Klammern sind notwendig, weil AND stärker bindet als OR.
-- (vgl. 'Punkt vor Strich') => Andere Ergebnismenge!
-- ************************************************************

SELECT "Artikel-Nr", "Bezeichnung", "H-Nr" FROM "tblArtikel"
WHERE ("Bezeichnung" = 'Bohrmaschine' OR "Bezeichnung" = 'Hammer')
        AND NOT "H-Nr" = '40103';

-- *****************************************
-- Gleiche Ergebnisse mit alternativen
-- Schreibweisen des Booleschen Operators NICHT.
-- *****************************************
-- so:
--         AND "H-Nr" <> '40103';
-- oder so:
--         AND "H-Nr" != '40103';
```

Artikel-Nr	Bezeichnung	H-Nr
101	Bohrmaschine	40122
121	Hammer	40122

Abb. 9: Ergebnis Beispiel 6

Beispiel 7 (*select-where-like-%.txt)*

```
-- ************************************
-- Liefert alle Datensätze aus tblArtikel,
-- mit der Teilzeichenkette 'Schrauben'.
-- Das ist z. B. gut, wenn du nicht weißt, ob
-- das Werkzeug als Schraubendreher oder
-- Schraubenzieher in der Datenbank vorliegt.
-- ************************************

SELECT "Artikel-Nr", "Bezeichnung" FROM "tblArtikel"
WHERE "Bezeichnung" LIKE 'Schrauben%';
```

Artikel-Nr	Bezeichnung
117	Schraubendreher

Abb. 10: Ergebnis Beispiel 7

Beispiel 8 (*select-where-between.txt)*

```
-- ************************************
-- Liefert alle Datensätze aus tblArtikel,
-- deren Wert im Feld 'Bestand' zwischen einem
-- Minimal- und einem Maximalwert liegen!
-- (MIN-Wert und MAX-Wert eingeschlossen.)
-- ************************************

SELECT "Artikel-Nr", "Bezeichnung", "Bestand" FROM "tblArtikel"
WHERE "Bestand" BETWEEN 75 AND 150;
```

Artikel-Nr	Bezeichnung	Bestand
117	Schraubendreher	84
121	Hammer	150

Abb. 11: Ergebnismenge Beispiel 6

Wie man die SQL-Anweisungen SELECT FROM, WHERE und ORDER BY in den Datenbanken Werkzeuge und RC Wildbach einsetzt, kannst du mit Hilfe dieser **Lernvideos** nachvollziehen:

Abb. 12: Startbild des Videotutoriums *SQL Teil 2*

Abb. 13: Startbild des Videotutoriums *SQL Teil 3*

Hinweis
In LibreOffice Base werden alle SELECT-Anweisungen direkt bei den Abfragen eingegeben (Abfragen >> Abfrage in SQL-Ansicht erstellen).

Alle anderen SQL-Befehle hingegen funktionieren nur im SQL-Fenster, das du unter dem Menüpunkt Extras>>SQL findest.