SQL-Befehlsliste

Vereinbarung über die Schreibweise

	Beschreibung	Beispiele
Schlüsselwort	Befehlsworte in SQL-Anweisungen werden fett und in Großbuchstaben geschrieben	**SELECT ... FROM ... ;**
[optionale Elemente]	mögliche, aber nicht zwingend erforderliche Teile von Anweisungen stehen [in eckigen Klammern]	[**WHERE** *Bedingung*]
	Kursiv gesetzte Begriffe stehen stellvertretend für die richtigen Namen von Datenbanken, Tabellen, Attributen	

SQL-Befehle

SQL	Beschreibung/Syntax/Beispiel

ALTER TABLE ...	verändert eine Tabellenstruktur
	Spalte (Attribut) hinzufügen:
	Syntax: ALTER TABLE *Tabellenname* ADD *Attributsname Datentyp*;
	z. B.: ALTER TABLE buecher ADD seitenanzahl INT;
	Spalte (Attribut) ändern:
	Syntax: ALTER TABLE *Tabellenname* MODIFY *Attributsname Datentyp*;
	z. B.: ALTER TABLE buecher MODIFY ort CHAR(50);
	Spalte (Attribut) löschen:
	Syntax: ALTER TABLE *Tabellenname* DROP *Attributsname*;
	z. B.: ALTER TABLE buecher DROP ort;
	Primärschlüssel hinzufügen:
	Syntax: ALTER TABLE *Tabellenname* ADD PRIMARY KEY (nummer);
	Fremdschlüssel hinzufügen:
	Syntax: ALTER TABLE *Tabellenname* ADD FOREIGN KEY (*Schlüsselfeld*) REFERENCES *Mastertabelle*(*Primärschlüssel*);
	z. B.: ALTER TABLE konten ADD FOREIGN KEY (nummer) REFERENCES banken(nummer);
	Primärschlüssel löschen:
	Syntax: ALTER TABLE *Tabellenname* DROP PRIMARY KEY;

CREATE DATABASE ...	erstellt eine Datenbank Syntax: CREATE DATABASE [IF NOT EXISTS] *Datenbankname*; z. B.: CREATE DATABASE IF NOT EXISTS fahrradvermietung;

CREATE TABLE ...	erstellt eine Tabelle Einfache Syntax: CREATE TABLE *Tabellenname* (*Attribut1 Datentyp*, *Attribut2 Datentyp*,); Vollständige Syntax: CREATE TABLE *Tabellenname* (*Attribut1 Datentyp*, *Attribut2 Datentyp*, [PRIMARY KEY (*Datenfeld*),] [FOREIGN KEY (*Datenfeld*) REFERENCES *Datenbankname.Tabellenname* (*Datenfeld*)] [ON DELETE {RESTRICT | CASCADE | SET NULL | NO ACTION}] [ON UPDATE {RESTRICT | CASCADE | SET NULL | NO ACTION}]) [ENGINE = INNODB] ; **PRIMARY KEY (***Attributsname***)** weist einem Attribut (Datenfeld) die Eigenschaft „Primärschlüssel“ zu. Syntax: PRIMARY KEY (*Attributsname*) **FOREIGN KEY (***Attributsname***)** weist einem Datenfeld die Eigenschaft „Fremdschlüssel“ zu und verweist („REFERENCES“) auf die Tabelle mit dem dazu gehörenden Primärschlüssel Syntax: FOREIGN KEY (*Attributsname*) REFERENCES *Tabellenname* (*Attributsname*)
DELETE FROM	löscht Datensätze in einer oder mehreren Tabellen einer Datenbank Syntax: DELETE FROM *Tabelle* WHERE *Bedingung*; z. B.: DELETE FROM kunden WHERE kunden.kdnr = "1241";
DESCRIBE	zeigt die Tabellenstruktur an Syntax: DESCRIBE *Tabellenname;*
DROP DATABASE	löscht die angegebene Datenbank Syntax: DROP DATABASE *Datenbankname*;
DROP TABLE	löscht die angegebene Tabelle Syntax: DROP TABLE *Tabellenname*;
INSERT INTO	fügt einer Tabelle einen (oder mehrere) Datensätze hinzu Syntax: **INSERT INTO** *Tabelle* [(*Spalte*[,*Spalte*,..)] **VALUES** (*Wert*[,*Wert*,...]; z. B.: INSERT INTO kunden (name, vorname) VALUES ("Maier", "Rudolf");

SELECT... FROM

Auswahl von Datensätzen und Feldern aus einer Datenbank

Syntax: SELECT [ALL/DISTINCT] {*Spalten*/*} FROM *Tabelle* [,*Tabelle2*,...]

z. B.: SELECT teile.teilenr, teile.bezeichnung FROM teile;
[WHERE *Bedingung*]
[GROUP BY *Spalten* [HAVING *Bedingung*]]
[ORDER BY *Spalten* [ASC/DESC]];

Erklärung: DISTINCT vermeidet die Auswahl doppelter/identischer Datensätze

AS gibt einer Spalte eine neue Überschrift

z. B.: SELECT mwsts AS Mehrwertsteuersatz FROM ...

BETWEEN ... AND

bestimmt, ob der Wert eines Ausdrucks innerhalb eines bestimmten Bereichs von Werten liegt.

z. B.: SELECT * FROM teile WHERE kpreis BETWEEN 20 AND 50;

GROUP BY

fasst Datensätze, die in der angegebenen Feldliste dieselben Werte enthalten, zu einem einzelnen Datensatz zusammen. Für jeden Datensatz wird ein Ergebniswert berechnet, wenn Sie eine SQL-Aggregatfunktion wie Sum oder Count in der SELECT-Anweisung angeben.

z. B.: SELECT plz, SUM(umsatz) AS Umsätze FROM lieferer GROUP BY plz;

Achtung: Dieser Befehlsteil muss immer der letzte Teil der SELECT-Anweisung sein!

HAVING

gibt an, welche der gruppierten Datensätze in einer SELECT-Anweisung mit einem GROUP BY-Abschnitt angezeigt werden sollen. Nachdem GROUP BY Datensätze kombiniert, zeigt HAVING alle vom GROUP BY-Abschnitt gruppierten Datensätze an, die die im HAVING-Abschnitt angegebenen Bedingungen erfüllen.

z. B.: ... FROM lieferer GROUP BY plz HAVING SUM(umsatz) > 30000;

INNER JOIN ... ON

kombiniert Datensätze aus zwei Tabellen, sobald ein gemeinsames Feld dieselben Werte hat.

Syntax: **FROM** *Tabelle1* **INNER JOIN** *Tabelle2* **ON** *Tabelle1.Feld1 = Tabelle2.Feld2*

z. B.: **FROM** bestpos **INNER JOIN** teile **ON** bestpos.teilenr = teile.teilenr;

WHERE

Der Teil einer SQL-Anweisung, der angibt, welche Datensätze abgerufen werden sollen. Der WHERE-Abschnitt beschränkt den Umfang der Abfrage (Selektion).

z. B.: SELECT * FROM teile WHERE teileart = "H";
SELECT * FROM artikel WHERE WG IS NULL;

IN

gibt die Datensätze zurück, die in dem in der Abfrage genannten Feld einen aus einer Auflistung von Werten beinhalten.

z. B.: ... WHERE teile.teileart IN ("E","T","R");

LIKE

dient zum Vergleichen zweier Zeichenfolgen.

z. B.: SELECT name FROM lieferer WHERE plz LIKE "7%";
(alle Lieferer-Datensätze des Postleitbereichs 7 werden ausgewählt)

ORDER BY

sortiert die Daten eines Recordset-Objekts nach einem oder mehreren angegebenen Feldern in aufsteigender oder absteigender Reihenfolge.

z. B.: SELECT liefnr, name FROM lieferer ORDER BY name ASC; (aufsteigend)

oder SELECT liefnr, name FROM lieferer ORDER BY name DESC; (absteigend)

SELECT ... INTO	erstellt eine neue Tabelle Syntax: SELECT {*Spalte*[,*Spalte*...]/*} INTO *Tabelle* FROM *Tabelle* [WHERE *Bedingung*]; z. B.: ;
SHOW DATABASES	listet alle vorhandenen Datenbanken auf Syntax: SHOW DATABASES;
UPDATE ... SET	ändert Werte in Feldern einer Tabelle Syntax: UPDATE *Tabelle* SET *Spalte = Ausdruck* [,*Spalte = Ausdruck* ...] [WHERE *Bedingung*]; z. B.: UPDATE teile SET ekpreis = ekpreis * 1.1; **Achtung: ohne WHERE-Klausel werden a l l e Datensätze geändert!**
USE	wählt eine Datenbank aus, die bearbeitet werden soll. Syntax: USE *Datenbankname*; z. B.: USE bibliothek;

SQL-Datentypen

numerische Datentypen	Beschreibung	Speicherplatz
DEC[(M,D)], DECIMAL[(M,D)] NUMERIC[(M,D)]	exakte Festkommazahl - mit M Stellen (ohne Dezimaltrennzeichen), - davon D Dezimalstellen	
DOUBLE[(M,D)], REAL[(M,D)]	Fließkommazahl mit doppelter Genauigkeit - mit M Stellen, - davon D Dezimalstellen	**8 Byte**
FLOAT[(M,D)]	Fließkommazahl mit einfacher Genauigkeit - mit M Stellen, - davon D Dezimalstellen	**4 Byte**
INT, INTEGER	ganze Zahlen - ohne Vorzeichen: von 0 bis 0 4.294.967.295 - mit Vorzeichen : von –2.147.483.648 bis 2.147.483.647 Nötiger Datentyp für AUTO_INCREMENT	**4 Byte**
String-Typen	**Beschreibung**	**Speicherplatz**
CHAR(M)	Zeichenkette mit fester Länge (M = 1 bis 255)	
VARCHAR(M)	Zeichenkette mit variabler Länge (M = 1 bis 65.532)	
Zeit-Datumstypen	**Beschreibung**	**Speicherplatz**
DATE	wenn Sie Werte benötigen, die nur das Datum enthalten: **'YYYY-MM-DD'**	
DATETIME	wenn Sie Werte benötigen, die sowohl ein Datum als auch eine Uhrzeit enthalten: **'YYYY-MM-DD HH:MM:SS'**	
YEAR	4stellige Jahresangabe; werden als Werte oder Zeichenkette behandelt	1 Byte
boolsche Typen	**Beschreibung**	**Speicherplatz**
TINYINT	wird als boolscher Datentyp behandelt. 0 bedeutet falsch (false), 1 oder >1 bedeutet wahr (true)	1 Byte
weitere Attribute:	**Beschreibung**	**Speicherplatz**
AUTO_INCREMENT	Der Wert dieses Attributs wird automatisch beim Anlegen eines neuen Datensatzes aus dem Wert des Datenfeldes des vorherigen Datensatzes + 1 berechnet. Nur bei INT-Typen	
DEFAULT (Wert)	Definiert einen Standardwert für dieses Feld	
NOT NULL	Die Eingabe eines Wertes für dieses Datenfeld wird erzwungen.	
NULL	Das Datenfeld hat standardmäßig keinen Wert.	

SQL-Funktionen

AVG()	berechnet den arithmetischen Mittelwert einer Menge von Werten in einem bestimmten Feld einer Abfrage. Syntax: AVG(Ausdr) z. B.: SELECT AVG(umsatz) FROM lieferer;
COUNT()	berechnet die Anzahl der von einer Abfrage zurückgegebenen Datensätze. Syntax: COUNT(Ausdruck) z. B.: SELECT COUNT(*) FROM teile WHERE ekpreis > 100;
CURRENT_DATE	liefert das Tagesdatum im Euro-Format: SELECT CURRENT_DATE *ergibt* 2018-01-01
DATE()	extrahiert den Datumsteil aus dem DATE- oder DATETIME-Ausdruck Syntax: DATE(Ausdruck) z. B.: SELECT * FROM auftrag WHERE aufdat = date(now());
DATEDIFF()	berechnet die Anzahl Tage zwischen dem Startdatum und dem Enddatum Syntax: DATEDIFF(Enddatum, Startdatum) z. B.: SELECT DATEDIFF(von, bis) FROM vertrag;
MAX()	gibt den größten Wert aus einer Reihe von Werten zurück, die in einem bestimmten Feld einer Abfrage enthalten sind. Syntax: MAX(Ausdr) z. B.: SELECT MAX(umsatz) AS hoechster_Umsatz FROM Lieferer;
MIN()	gibt den kleinsten Wert aus einer Reihe von Werten zurück, die in einem bestimmten Feld einer Abfrage enthalten sind. Syntax: MIN(Ausdr) z. B.: SELECT MIN(umsatz) AS kleinster_Umsatz FROM Lieferer;
NOW()	liefert das aktuelle Tagesdatum mit Uhrzeit Syntax: NOW() z. B.: SELECT * FROM auftrag WHERE aufdat = now();
SUM()	berechnet die Summe einer Menge von Werten in einem bestimmten Feld einer Abfrage. z. B.: SELECT SUM(umsatz) AS liefererumsaetze FROM Lieferer;
TIMEDIFF()	berechnet den Zeitraum zwischen der Startzeit datum1 und der Endzeit datum2. Syntax: TIMEDIFF(datum1, datum2) Ausgabe in der Form HH:MM:SS, z. B. 12:40:33 oder 55:12:20. Maximal möglich sind 838:59:59
YEAR()	liefert das Jahr eines Datums; kann benutzt werden, um die Jahresdifferenz auszurechnen z. B. SELECT year(now()) - year(gebdat) FROM teilnehmer;

Logische Operatoren

AND	verknüpft zwei Bedingungen, die beide gelten müssen z. B. WHERE nachname = 'Kurz' AND ort = 'Berlin';
OR	verknüpft zwei Bedingungen, von denen nur eine gelten muss z. B. WHERE nachname = 'Kurz' OR ort = 'Berlin';

Nachschlagemöglichkeiten:

http://sql.1keydata.com/de/