Kurzreferenz SQL-Befehle (MySQL, MariaDB)

In diesem Dokument wird die Syntax wichtiger SQL-Befehle aufgelistet. Die Syntax der Statements weicht in jedem DBS ab. Bitte lesen Sie deshalb die Dokumentation des jeweiligen DBS, um alle Möglichkeiten auszuschöpfen.

Anmerkungen zur Syntax:

Folgende Zeichen sind nicht Bestandteil der SQL-Statements: **[], {}, |**

[...]: Was in eckigen Klammern steht, ist optional und kann auch weggelassen werden.

{Option A|Option B|...}: Eine Option in diesen Klammern muss gewählt werden.

... oder ,...: Drei Punkte oder ein Komma mit drei Punkten symbolisieren eine beliebige Wiederholung der vorherigen Struktur.

Datenbank-Befehle

```
CREATE DATABASE [IF NOT EXISTS] db_name;
DROP DATABASE [IF EXISTS] db_name;
SHOW DATABASES;
USE db_name;
```

Daten abfragen

```
SELECT [ALL | DISTINCT | DISTINCTROW ] select_expression,...
[FROM table_references
[WHERE where_condition]
[GROUP BY {col_name|expression|position}, ... [WITH ROLLUP]]
[HAVING where_condition]
[ORDER BY {col_name|expression|position} [ASC|DESC],...[WITH ROLLUP]];
```

select_expression: {column|expression} [[AS] alias_name]

col_name: Spaltenname

expression: Ausdruck der Spalten und Funktionen enthalten kann

position: Nummer der Position des select_expression im SELECT-Abschnitt

table_references: {single_table|joined_table}

single_table: {tbl_name [[AS] alias_name]}

joined_table:
```
{
  table_name {[INNER|CROSS]} JOIN table_name ON search_condition
  |table_name {[LEFT|RIGHT]} [OUTER] JOIN table_name ON search_condition
}
```

where_condition:
Kriterien, die ein Datensatz erfüllen muss, um ausgewählt zu werden.
z.B. col_name1 > 10 [{OR | AND} col_name_2 = 100] ...

search_condition:
Spalten, die miteinander verbunden werden.
z.B. tab1.col_name1 = tab2.col_name2 [AND ...]

WITH ROLLUP: Berechnet Gesamtsummen für jede Ebene der Gruppierung.

Daten bearbeiten

```
INSERT INTO tbl_name [(col_name [, col_name] ...)]
    {VALUES | VALUE} (value_list) [, (value_list)] ...;

INSERT INTO tbl_name [(col_name [, col_name] ...)]
    {SELECT ... | TABLE table_name};

    value_list: value [,value]

    value: {expression | DEFAULT}

UPDATE  tbl_name SET assignment_list [WHERE where_condition];

    assignment_list: assignment [, assignment] ...

    assignment: col_name = value

    value: {expression | DEFAULT}


DELETE FROM tbl_name [[AS] tbl_alias]
    [WHERE where_condition];

    where_condition:
    Kriterien, die ein Datensatz erfüllen muss, um ausgewählt zu werden.
         z.B. col_name1 > 10 [{OR | AND} col_name_2 = 100] ...
```

Mit Tabellen arbeiten

```
ALTER TABLE tbl_name
    [alter_option [, alter_option] ...];

    alter_option:
    {   ADD [COLUMN] col_name column_definition
      | ADD [CONSTRAINT [constraint_name]] PRIMARY KEY
            (key_part,...)
      | ADD [CONSTRAINT [constraint_name]] UNIQUE [INDEX | KEY]
            [index_name] (key_part,...)
      | ADD [CONSTRAINT [symbol]] FOREIGN KEY
            [index_name] (col_name,...)
            reference_definition
      | DROP {CHECK | CONSTRAINT} constraint_name
      | ALTER [COLUMN] col_name
            {SET DEFAULT {literal|(expression)|DROP DEFAULT}
      | CHANGE [COLUMN] old_col_name new_col_name column_definition
      | DROP [COLUMN] col_name
      | DROP PRIMARY KEY
      | DROP FOREIGN KEY constraint_name
      | MODIFY [COLUMN] col_name column_definition [FIRST | AFTER col_name]
      | RENAME COLUMN old_col_name TO new_col_name
      | RENAME [TO | AS] new_tbl_name
    }

    column_definition: {
        data_type [NOT NULL | NULL] [DEFAULT {literal | (expression)} ]
        [AUTO_INCREMENT] [UNIQUE [KEY]] [[PRIMARY] KEY]
        [COMMENT 'string']
    }
    key_part: {col_name | (expression)}
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name (create_definition,...);
```

```
create_definition: {
    col_name column_definition
  | [CONSTRAINT [constraint_name]] PRIMARY KEY (key_part,...)
  | [CONSTRAINT [constraint_name]] UNIQUE [INDEX | KEY] (key_part,...)
  | [CONSTRAINT [constraint_name]] FOREIGN KEY  (col_name,...)
     reference_definition
  | check_constraint_definition
}

column_definition: {
    data_type [NOT NULL | NULL] [DEFAULT {literal | (expression)} ]
    [AUTO_INCREMENT] [UNIQUE [KEY]] [[PRIMARY] KEY]
    [COMMENT 'string']
}

data_type: Verwenden Sie die Datentyp-Liste

key_part: {col_name| (expression)}

check_constraint_definition:
    [CONSTRAINT [symbol]] CHECK (expression) [[NOT] ENFORCED]

reference_definition:
    REFERENCES tbl_name (key_part,...)
```

```
DROP TABLE [IF EXISTS] tbl_name;

SHOW TABLES [FROM db_name][LIKE term_with_wildcard];
```

term_with_wildcard: Begriff der Platzhalter enthält - z.B. **%** (beliebig viele Zeichen) oder **_** (ein Zeichen)

Mit Views arbeiten

```
CREATE [OR REPLACE] VIEW view_name AS select_statement
[WITH CHECK OPTION];
```

Mit Triggern arbeiten

```
CREATE TRIGGER [IF NOT EXISTS] trigger_name trigger_time trigger_event ON tbl_name
FOR EACH ROW trigger_body

trigger_time: {BEFORE | AFTER}
trigger_event: {INSERT | UPDATE | DELETE}

DROP TRIGGER [IF EXISTS] trigger_name
```

Trennzeichen einstellen

```
delimiter ||
```