Syntaxbeschreibungen anwenden

LES

2021-01-11

Für Computerbefehle gibt es eine kompakte Beschreibung der Syntax (Rechtschreibung und Grammatik), die wie folgt aussieht:

> **SELECT** [**ALL**|**DISTINCT**] {*Spalten*|*} **FROM** *Tabelle* [,*Tabelle, …*]
> [**WHERE** *Bedingung*]
> [**GROUP BY** *Spalten* [**HAVING** *Bedingung*]]
> [**ORDER BY** *Spalten*] [**ASC**|**DESC**]

1 Erklärung

- **fett** gedruckt sind SQL-Schlüsselwörter. Sie müssen genau in der angegebenen Reihenfolge übernommen werden, Groß- und Kleinschreibung ist bei SQL-Schlüsselwörtern egal.
- *kursiv* gedruckt sind Platzhalter. Dafür setzen Sie Namen von Datenbankobjekten oder etwas selbst Formuliertes ein.
  Spalten steht für den Namen einer Spalte oder für mehrere Spalten, die Sie mit Komma getrennt aufzählen.
- *<Spitze Klammern>* Was in spitzen Klammern steht, wird woanders definiert.
- [eckige Klammern] kennzeichnen optionale Teile, das heißt: Was in eckigen Klammern steht, müssen Sie nicht verwenden. Beachten Sie die Verschachtelung: `HAVING` dürfen Sie nur verwenden, wenn Sie auch `GROUP BY` verwendet haben.
- Ein senkrechter Strich | kennzeichnet Alternativen. Umliegende eckige oder geschweifte Klammern { } grenzen den Bereich der Alternativen ab. Sie zählen also entweder die Spalten auf oder Sie verwenden ein * für alle Spalten; Sie können das Schlüsselwort `ALL` oder das Schlüsselwort `DISTINCT` eintippen, aber nicht beide (Sie können auch keines von beiden eintippen, da die beiden in eckigen Klammern [ ] stehen) – das gleiche gilt für `ASC` und `DESC`.
- Klammern und Striche werden nicht eingetippt.

2 Beispiele

Sie haben folgende Tabelle, die Sie abfragen möchten:

Kunde

Kundennummer	*Firmenname*	*PLZ*	*Ort*
1	Müller	75175	Pforzheim
2	Maier	75249	Kieselbronn
3	Schulze	76012	Karlsruhe
4	Fischer	75177	Pforzheim

1. Sie möchten alle Spalten und alle Zeilen sehen:
   `SELECT * FROM Kunde`
   Der Ausdruck *FROM Kunde* macht klar, dass die Tabelle *Kunde* gelesen werden soll. Das Sternchen * steht für alle Spalten. Da Sie nicht `ORDER BY …` sagen, ist die Reihenfolge der Zeilen unbestimmt.

2. Sie möchten nur die Orte sehen:
   `SELECT Ort FROM Kunde`
   Der Name einer Spalte ersetzt das Sternchen.

3. Sie möchten nur Postleitzahlen und Orte sehen:
   `SELECT PLZ, Ort FROM Kunde`
   Mehrere Spalten werden mit Komma getrennt aufgezählt.

4. Sie möchten die Orte sehen, aber jeden Ort nur einmal:
   `SELECT DISTINCT Ort FROM Kunde`
   DISTINCT entfernt Duplikate.

5. Sie möchten die Kundenliste nach Namen sortiert:
   `SELECT * FROM Kunde ORDER BY Firmenname`
   (also von Fischer bis Schulze)

6. Sie möchten die Kundenliste nach Ort absteigend sortiert:
   `SELECT * FROM Kunde ORDER BY Ort DESC`
   (zuerst zwei Kunden aus Pforzheim, dann der aus Kieselbronn, am Schluss der aus Karlsruhe; die Reihenfolge der Kunden aus Pforzheim ist unbestimmt.)

7. Sie möchten die Kundenliste nach Ort und dann (bei gleichem Ort) nach Namen sortiert:
   `SELECT * FROM Kunde ORDER BY Ort, Firmenname`
   (also in Pforzheim zuerst Fischer, dann Müller)

8. Sie interessieren sich nur für Kunde 3:
   `SELECT * FROM Kunde WHERE Kundennummer = 3`

9. Sie interessieren sich nur für Firma Fischer:
   `SELECT * FROM Kunde WHERE Firmenname = 'Fischer'`
   Text muss immer in Hochkommas. Vorsicht: Innerhalb von Hochkommas wird Groß- und Kleinschreibung beachtet – 'Fischer' ≠ 'fischer' ≠ 'FISCHER'.

10. Sie interessieren sich für Fischer, egal ob groß- oder kleingeschrieben:
    `SELECT * FROM Kunde WHERE UPPER(Firmenname) = 'FISCHER'`
    Die Zeilenfunktion `UPPER` wandelt in Großbuchstaben um.

11. Sie suchen Kunden, deren Name mit einem großen M anfängt:
    `SELECT * FROM Kunde WHERE Firmenname LIKE 'M%'`
    LIKE macht einen Mustervergleich, das Prozentzeichen `%` steht für beliebig viele Zeichen, der Unterstrich `_` für ein einziges.

12. Sie wissen nicht, ob sich Maier mit ai, ei, ay oder ey schreibt:
    `SELECT * FROM Kunde WHERE Firmenname LIKE 'M__er'`

13. Sie möchten wissen, wieviele Zeilen die Tabelle hat:
    `SELECT COUNT(*) FROM Kunde`
    COUNT ist eine Spaltenfunktion (Aggregatfunktion).

14. Sie möchten wissen, wieviele Kunden in jedem Ort sitzen:
    `SELECT Ort, COUNT(*) FROM Kunde GROUP BY Ort`

15. Sie möchten wissen, wieviele Kunden in jedem Ort sitzen, aber nur für Orte mit mehr als einem Kunden:

```
SELECT Ort, COUNT(*) FROM Kunde
GROUP BY Ort HAVING COUNT(*) > 1
```

Im Unterschied zu `WHERE` wird die Bedingung nach `HAVING` nicht direkt auf die Tabelle angewendet, sondern auf das Ergebnis der Spaltenfunktion und Gruppierung.

3 Befehlsübersicht Datendefinition (SQL-DDL)

Das Anlegen einer Datenbank `CREATE DATABASE` ist *kein* Bestandteil von SQL. Wegen Besonderheiten einiger Systeme hilft nur ein Blick in das Handbuch des Datenbanksystems. Die folgenden Angaben sind stark gekürzt, benutzen Sie ein Handbuch, wenn Sie Besonderheiten benötigen.

CREATE TABLE *Tabellenname* **(**
<Spaltendefinition> [,*<Spaltendefinition>, ...*]
[**PRIMARY KEY (** *Spaltenname* [*, Spaltenname ...*] **)**]
[**FOREIGN KEY (***Spaltenname***) REFERENCES** *Tabellenname***(***Spaltenname***)**]
)

`CREATE TABLE` erzeugt eine Tabelle. *Tabellennamen* beginnen mit einem Buchstaben und können Buchstaben und Ziffern enthalten. Vermeiden Sie Namen, die SQL-Schlüsselwörter sind. Vermeiden Sie Sonderzeichen. Groß- und Kleinschreibung ist normalerweise egal – außer, wenn Sie den Namen in doppelte Anführungszeichen setzen[1].

`<Spaltendefinition> ::=`
Spaltenname Datentyp
[**NOT NULL**]
[**DEFAULT** *Standardwert*]
[**PRIMARY KEY**] **)** [**FOREIGN KEY REFERENCES** *Tabelle***(***Spalte***)**]

Für die Auswahl von *Spaltennamen* gilt das, was auch bei Tabellennamen gesagt wurde. Datentyp ist zum Beispiel `INTEGER`, `DOUBLE` oder `VARCHAR(40)`.

NOT NULL schaltet die dreiwertige Logik ab. *DEFAULT Standardwert* setzt statt NULL einen Standardwert, wenn beim Einfügen nichts angegeben wurde. `PRIMARY KEY` macht die Spalte zur Primärschlüsselspalte. Falls Ihr Primärschlüssel zusammengesetzt ist, verwenden Sie statt dessen eine Primärschlüsseldefinition am Schluss des *CREATE TABLE* Statements. **FOREIGN KEY REFERENCES** stellt die referentielle Integrität sicher: Ein Wert kann nur eingetragen werden, wenn er in der referenzierten Tabelle existiert. Sie können die Fremdschlüsselbedingungen auch am Schluss der Tabellendefinition machen, dann muss aber der Spaltenname mit dazu.

Sie können nach den Schlüsselworten **PRIMARY KEY** die Spalte(n) aufzählen, die den Primärschlüssel bilden, falls Ihr Primärschlüssel zusammengesetzt ist, z. B. `... PRIMARY KEY (Kursnummer, Teilnehmernummer)`.

```
CREATE TABLE ARTIKEL (ARTIKELNR INTEGER NOT NULL PRIMARY KEY,
ARTIKELNAME VARCHAR(40) NOT NULL, GTIN CHAR(14),
PREIS DECIMAL(7,2) NOT NULL, WARENGRUPPE INTEGER NOT NULL
FOREIGN KEY REFERENCES ARTIKELGRUPPE(GRUPPENNR),
BESCHREIBUNG VARCHAR(200))
```

Eine Tabelle namens Artikel wird angelegt. Sie hat folgende Spalten:

1. Artikelnr – ganze Zahl, gleichzeitig Primärschlüssel,

[1] oder MYSQL verwenden. Schreiben Sie dort lieber alle Tabellennamen klein

2. Artikelname – Text mit bis zu 40 Anschlägen, Pflicht,
3. GTIN – Text mit genau 14 Anschlägen, keine Pflicht,
4. Preis – Zahl mit 5 Stellen vor und 2 Stellen nach dem Komma,
5. Warengruppe – ganze Zahl, Pflicht, muss aus dem Kreis der Nummern in Spalte *Gruppennr* der Tabelle *Artikelgruppe* kommen,
6. Beschreibung – Text mit bis zu 200 Anschlägen.

DROP TABLE *Tabellenname*

Dieser Befehl löscht eine Tabelle. Wenn Sie die Tabelle nur leeren möchten, verwenden Sie `DELETE FROM`.

ALTER TABLE *Tabellenname* **ADD COLUMN** *Spaltendefinition* [*, Spaltendefinition …*]

So fügen Sie einer Tabelle weitere Spalten an. Wenn die Tabelle bereits Daten enthält, können Sie NULL-Werte für die neue Spalte nicht ausschließen.

ALTER TABLE *Tabellenname* **DROP COLUMN** *Spaltenname* [*, Spaltenname …*]

So löschen Sie Spalten aus einer Tabelle.

ALTER TABLE *Tabellenname* **MODIFY COLUMN** *Spaltenname datentyp*

ALTER TABLE *Tabellenname* **ADD CONSTRAINT** *Fremdschlüsselbedingung*

Das einzige, was zuverlässig bei `ALTER TABLE` funktioniert, ist die Erhöhung der maximalen Breite einer VARCHAR-Spalte. Alles andere ist abhängig von den bereits in der Tabelle liegenden Daten, zudem systemabhängig und damit Glückssache.

CREATE INDEX *Indexname* **ON** *Tabellenname***(***Spaltenname* [*, Spaltenname …*] **)**

Zur Beschleunigung von Lesezugriffen können Sie einen Index anlegen. Schreibende Zugriffe werden dadurch allerdings langsamer und der Platzbedarf nimmt zu.

Beispiel: `CREATE INDEX ARTIKELGTIN ON ARTIKEL(GTIN)`

Für die Tabelle *Artikel* wird ein Index über die Spalte *GTIN* erstellt.

DROP INDEX *Indexname*

So löschen Sie einen Index. Mit `EXPLAIN` können Sie herausfinden, ob ein Index sinnvoll ist, um eine bestimmte Abfrage zu beschleunigen.

CREATE VIEW *Viewname* **AS** *SELECT-Ausdruck*

So können Sie eine Abfrage dauerhaft als View mit einem Namen in der Datenbank ablegen. Ein View kann in vielen Fällen wie eine Tabelle benutzt werden.