stackfuel

SQL Codeübersicht - Kapitel 1

Hier findest du die wichtigsten Befehle der Lektionen mit Beispielcode.
Wenn du dir den Output jeder Query noch einmal anschauen willst, kannst du einen Blick in die Notebooks werfen.

Datenbankstruktur einsehen

`SHOW` zeigt dir alle Tabellen einer Datenbank:

```
SHOW tables;
```

`DESC` gibt dir die Übersicht über eine Tabelle:

```
DESC <tablename>;
```

Mit `DESC customer;` erhältst du beispielsweise folgende Übersicht:

Field	Type	Null	Key	Default	Extra
customer_id	smallint unsigned	NO	PRI	None	auto_increment
store_id	tinyint unsigned	NO	MUL	None	
first_name	varchar(45)	NO		None	
last_name	varchar(45)	NO	MUL	None	
email	varchar(50)	YES		None	
address_id	smallint unsigned	NO	MUL	None	
active	tinyint(1)	NO		1	
create_date	datetime	NO		None	
last_update	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED on update CURRENT_TIMESTAMP

Field: Spaltenname
Type: Datentyp der Spalte
Null: Fehlende Einträge in der Spalte erlaubt (YES) oder nicht (NO)
Key: ob die Spalte Teil eines Schlüssels ist und falls ja, welcher Art von Schlüssel
Default: Standardwert dieser Spalte
Extra: sonstige Informationen

stackfuel

SELECT

Um Daten aus Tabellen abzufragen, wird in SQL stets die `SELECT`-Anweisung verwendet.
Angegeben werden immer zuerst die Spalten und anschließend aus welcher Tabelle diese stammen.

```
SELECT first_name FROM staff;
```

Mehrere Spalten:

```
SELECT first name,
       last name,
FROM staff;
```

Mehrere Spalten:

```
SELECT *
FROM staff;
```

Spalten können direkt mit Expressions bearbeitet werden:

```
SELECT amount * 0.85
FROM payment;
```

... und einen Alias erhalten:

```
SELECT amount AS amount_dollar,
       amount * 0.85
              AS amount_euro
FROM payment;
```

Informationsfunktionen benötigen keine Tabellen:

```
SELECT user(),
       database(),
       version();
```

Andere Funktionen werden auf Tabellen angewandt:

```
SELECT DAYNAME(payment_date)
       AS weekday
FROM payment;
```

LIMIT

Beschränkt die Ausgabe einer Query auf n Zeilen (hier 5):

```
SELECT first_name, last_name
FROM customer
LIMIT 5;
```

Mit Komma werden m Zeilen übersprungen und n Zeilen angezeigt (hier m=10, n=5)

```
SELECT first_name, last_name
FROM customer
LIMIT 10,5;
```

DISTINCT

Beschränkt die Ausgabe auf einzigartige Werte (hier einzigartige store_ids):

```
SELECT DISTINCT store_id
FROM inventory;
```

Achtung: `DISTINCT` sollte bei Tabellen mit vielen Zeilen bedacht eingesetzt werden, da die Ergebnisse zum Auffinden und Entfernen der Duplikate sortiert werden. Daten in einer relationalen Datenbank sind unsortiert. Das Sortieren großer Datenmengen kann viel Zeit kosten.

stackfuel

Aggregationsfunktionen

Die wichtigsten Aggregationen sind `AVG()`, `MIN()`, `MAX()`, `COUNT()`:

```
SELECT AVG(amount)
       AS amount_average
FROM payment;
```

Sie können mit `DISTINCT` kombiniert werden:

```
SELECT COUNT(amount),
       COUNT(DISTINCT amount)
FROM payment;
```

Zeit- und Datumsfunktionen

Diese Funktionen erleichtern das Arbeiten mit Zeitstempeln und Datumsangaben. Es gibt unter anderem `DATE()`, `YEAR()`, `MONTH()`, `DAYNAME()`, `HOUR()`, `MINUTE()`, `SECOND()`:

```
SELECT DAYNAME(payment_date) AS weekday,
       HOUR(payment_date) AS hour
FROM payment;
```

WHERE

Filtert die ausgewählten Daten nach bestimmten Merkmalen. Hierfür werden Vergleichsoperatoren (z.B. `=`, `>`, `<=`, `IN`, `LIKE`, `BETWEEN … AND …`) und logische Operatoren wie `AND`, `OR` verwendet:

```
SELECT *
FROM payment
WHERE YEAR(payment_date) = 2005;

SELECT *
FROM payment
WHERE YEAR(payment_date) >= 2005
      AND MONTH(payment_date) BETWEEN 6 AND 9;

SELECT *
FROM payment
WHERE customer_id = 84
      OR customer_id IN (203, 204, 207);
```