stackfuel

SQL Codeübersicht - Kapitel 2

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.

SQL-Abfragen folgen immer diesem Muster:

1. `SELECT` Spalten in Ergebnistabelle
2. `FROM` Tabelle
   2a. `JOIN`
3. `WHERE` Filterausdruck mit Bedingungen
4. `GROUP BY` Spalten, nach deren Werten gruppiert werden soll
5. `HAVING` Filterausdruck mit Bedingungen für aggregierte/gruppierte Spalten
6. `ORDER BY` Sortierkriterien, optional: `DESC`
7. `LIMIT`

GROUP BY

Gruppiert die Datensätze anhand einer oder mehrerer Spalten (z.B. für Aggregationsfunktionen)

Einfach:

```
SELECT customer_id,
       COUNT(*) AS Payments
FROM payment
GROUP BY customer_id;
```

Mehrfach:

```
SELECT customer_id,
       staff_id,
       COUNT(*) AS Payments
FROM payment
GROUP BY customer_id, staff_id;
```

HAVING

Filtert Daten nach einer Gruppierung

```
SELECT customer_id,
       COUNT(*) AS Payments
FROM payment
WHERE customer_id BETWEEN 12 AND 50
GROUP BY customer_id
HAVING Payments >= 35;
```

ORDER BY

Sortiert Daten anhand einer oder mehrerer Spalten auf- (`ASC`) oder absteigend (`DESC`)

```
SELECT customer_id,
       COUNT(*) AS Payments
FROM payment
WHERE customer_id BETWEEN 12 AND 50
GROUP BY customer_id
HAVING Payments >= 35
ORDER BY Payments DESC;
```

stackfuel

JOIN (INNER JOIN)

Verbindet Tabellen anhand eines gemeinsamen Schlüssels (Keys):

```
SELECT *
FROM store
INNER JOIN address
   ON store.address_id = address.address_id;
```

`JOIN` kann genutzt werden, um Tabellen mit sich selbst zu verbinden *(self join)*, z.B. wenn die Daten aus verschiedenen Zeilen einer Spalte benötigt werden

```
SELECT a1.address AS store_address,
       a2.address AS manager_address
FROM store AS s
INNER JOIN staff AS st
    ON s.manager_staff_id = st.staff_id
INNER JOIN address AS a1
    ON s.address_id = a1.address_id
INNER JOIN address AS a2
    ON st.address_id = a2.address_id
WHERE s.store_id = 2;
```

LEFT JOIN, RIGHT JOIN (OUTER JOIN)

Verbindet Tabellen anhand eines gemeinsamen Schlüssels (Keys), wobei Tabelle 1 um die Werte von Tabelle 2 erweitert wird, wenn keine Werte existieren, wird `NULL` eingefügt:

```
SELECT *
FROM customer_1 AS c1
LEFT JOIN customer_2 AS c
    ON c1.customer_id = c2.customer_id;
```

CROSS JOIN

Verbindet Tabellen als kartesisches Produkt (alle möglichen Paare der Tabellenzeilen). Hierbei wird kein gemeinsamer Schlüssel (Keys) benötigt. Dieser Join ist sehr rechenintensiv und selten sinnvoll.

```
SELECT *
FROM customer_1
CROSS JOIN customer_2;
```

Subqueries

Subqueries folgen der normalen Query Struktur.
Achte darauf, sie in Klammern zu setzen und für `JOIN`s und `UNION`s zu benennen:

```
SELECT customer_id
FROM payment
WHERE amount = (SELECT MAX(amount)
                FROM payment);
```

stackfuel

UNION/UNION ALL

`UNION` verbindet zwei Tabellen miteinander. Dabei werden die Tabellen übereinander ‚gestapelt'. Beide Tabellen müssen dabei die gleichen Spalten enthalten. `UNION ALL` löscht zusätzlich Duplikate.

```
(SELECT c.customer_id, c.store_id,
       SUM(p.amount) AS total_payments
FROM customer AS c
INNER JOIN payment AS p
ON c.customer_id = p.customer_id
WHERE c.store_id = 1
GROUP BY c.customer_id
ORDER BY total_payments DESC
LIMIT 10)

UNION ALL

(SELECT c.customer_id, c.store_id,
        SUM(p.amount) AS total_payments
FROM customer AS c
INNER JOIN payment AS p
ON c.customer_id = p.customer_id
WHERE c.store_id = 2
GROUP BY c.customer_id
ORDER BY total_payments DESC
LIMIT 10)) AS full_table;
```