KAPITEL 10

# Wie mache ich ...?

Dieses Kapitel soll als schnelle Referenz für häufig gestellte SQL-Fragen dienen, die mehrere Konzepte miteinander kombinieren:

- Zeilen finden, die doppelte Werte enthalten
- Zeilen mit dem Maximalwert für eine andere Spalte auswählen
- Text aus mehreren Feldern in einem einzigen Feld verketten
- Alle Tabellen finden, die einen bestimmten Spaltennamen enthalten
- Eine Tabelle aktualisieren, deren ID einer anderen Tabelle entspricht

## Zeilen finden, die doppelte Werte enthalten

Die folgende Tabelle listet sieben Teesorten auf sowie die Temperaturen, bei denen sie ziehen sollten. Beachten Sie, dass es zwei Gruppen mit doppelten `tea`/`temperature`-Werten gibt; diese werden fett dargestellt.

```
SELECT * FROM teas;

+----+--------+-------------+
| id | tea    | temperature |
+----+--------+-------------+
|  1 | green  |         170 |
|  2 | black  |         200 |
|  3 | black  |         200 |
|  4 | herbal |         212 |
```

```
|  5 | herbal |         212 |
|  6 | herbal |         210 |
|  7 | oolong |         185 |
+----+--------+-------------+
```

Dieser Abschnitt behandelt zwei unterschiedliche Szenarien:

- Zurückliefern aller einmaligen tea/temperature-Kombinationen
- Zurückliefern nur der Zeilen mit doppelten tea/temperature-Werten

## Alle einmaligen Kombinationen zurückliefern

Um alle doppelt vorhandenen Werte auszuschließen und nur die einmaligen Zeilen einer Tabelle zurückzuliefern, verwenden Sie das Schlüsselwort DISTINCT.

```
SELECT DISTINCT tea, temperature
FROM teas;

+--------+-------------+
| tea    | temperature |
+--------+-------------+
| green  |         170 |
| black  |         200 |
| herbal |         212 |
| herbal |         210 |
| oolong |         185 |
+--------+-------------+
```

### Potenzielle Erweiterungen

Um die Anzahl der einmaligen Zeilen in einer Tabelle zurückzuliefern, setzen Sie die Schlüsselwörter COUNT und DISTINCT zusammen ein. Näheres finden Sie im Abschnitt »DISTINCT« in Kapitel 4.

## Nur die Zeilen mit doppelt vorhandenen Werten zurückliefern

Die folgende Abfrage identifiziert die Zeilen in der Tabelle, in denen doppelt vorhandene Werte stehen.

```
WITH dup_rows AS (
     SELECT tea, temperature,
            COUNT(*) as num_rows
     FROM teas
     GROUP BY tea, temperature
     HAVING COUNT(*) > 1)

SELECT t.id, d.tea, d.temperature
FROM teas t INNER JOIN dup_rows d
     ON t.tea = d.tea
     AND t.temperature = d.temperature;

+----+--------+-------------+
| id | tea    | temperature |
+----+--------+-------------+
|  2 | black  |         200 |
|  3 | black  |         200 |
|  4 | herbal |         212 |
|  5 | herbal |         212 |
+----+--------+-------------+
```

### Erklärung

Der größte Teil der Arbeit geschieht in der dup_rows-Abfrage. Alle tea/temperature-Kombinationen werden gezählt, und dann werden mit der HAVING-Klausel nur die Kombinationen aufgehoben, die mehr als einmal vorkommen. So sieht dup_rows aus:

```
+--------+-------------+----------+
| tea    | temperature | num_rows |
+--------+-------------+----------+
| black  |         200 |        2 |
| herbal |         212 |        2 |
+--------+-------------+----------+
```

Der Zweck des JOIN in der zweiten Hälfte der Abfrage ist, die id-Spalte zurück in die fertige Ausgabe zu ziehen.

### Schlüsselwörter in der Abfrage

- **WITH dup_rows** ist der Beginn einer Common Table Expression, die es Ihnen erlaubt, innerhalb einer einzigen Abfrage mit mehreren SELECT-Anweisungen zu arbeiten.

- **`HAVING COUNT(*) > 1`** nutzt die `HAVING`-Klausel, die es Ihnen erlaubt, auf einer Aggregation wie `COUNT()` zu filtern.
- **`teas t INNER JOIN dup_rows d`** verwendet einen `INNER JOIN`, der es Ihnen erlaubt, die `teas`-Tabelle und die `dup_rows`-Abfrage zusammenzuführen.

### Potenzielle Erweiterungen

Um bestimmte Zeilen aus einer Tabelle zu löschen, verwenden Sie eine `DELETE`-Anweisung. Näheres finden Sie in Kapitel 5.

## Zeilen mit einem Maximalwert für eine andere Spalte auswählen

Die folgende Tabelle listet Angestellte und die von ihnen ausgeführten Verkäufe auf. Sie wollen für jeden Angestellten die neueste Anzahl an Verkäufen zurückliefern. Dieser Wert ist jeweils fett gedruckt.

```
SELECT * FROM sales;

+------+----------+------------+-------+
| id   | employee | date       | sales |
+------+----------+------------+-------+
|    1 | Emma     | 2021-08-01 |     6 |
|    2 | Emma     | 2021-08-02 |    17 |
|    3 | Jack     | 2021-08-02 |    14 |
|    4 | Emma     | 2021-08-04 |    20 |
|    5 | Jack     | 2021-08-05 |     5 |
|    6 | Emma     | 2021-08-07 |     1 |
+------+----------+------------+-------+
```

### Lösung

Die folgende Abfrage liefert die Anzahl der Verkäufe zurück, die jeder Angestellte zum neuesten Verkaufsdatum ausgeführt hat (d.h. zum größten Datumswert jedes Verkäufers).

```
SELECT s.id, r.employee, r.recent_date, s.sales
FROM (SELECT employee, MAX(date) AS recent_date
      FROM sales
      GROUP BY employee) r
INNER JOIN sales s
           ON r.employee = s.employee
           AND r.recent_date = s.date;

+------+----------+-------------+-------+
| id   | employee | recent_date | sales |
+------+----------+-------------+-------+
|    5 | Jack     | 2021-08-05  |     5 |
|    6 | Emma     | 2021-08-07  |     1 |
+------+----------+-------------+-------+
```

## Erklärung

Der Schlüssel zu diesem Problem liegt darin, es in zwei Teile zu zerlegen. Das erste Ziel ist, für jeden Angestellten das neueste Verkaufsdatum zu identifizieren. So sieht die Ausgabe der Unterabfrage `r` aus:

```
+----------+-------------+
| employee | recent_date |
+----------+-------------+
| Emma     | 2021-08-07  |
| Jack     | 2021-08-05  |
+----------+-------------+
```

Das zweite Ziel ist, die Spalten `id` und `sales` zurück in das Endergebnis zu ziehen. Das wird durch den `JOIN` in der zweiten Hälfte der Abfrage erledigt.

## Schlüsselwörter in der Abfrage

- **`GROUP BY employee`** verwendet die `GROUP BY`-Klausel, die die Tabelle anhand von `employee` zerlegt und für jeden Angestellten **`MAX(date)`** sucht.
- **`r INNER JOIN sales s`** benutzt einen `INNER JOIN`, der es Ihnen erlaubt, die Unterabfrage `r` und die `sales`-Tabelle zusammenzuführen.

#### Potenzielle Erweiterungen

Eine Alternative zur GROUP BY-Lösung besteht darin, eine Fensterfunktion (OVER ... PARTITION BY ...) mit einer FIRST_VALUE-Funktion zu verwenden, die die gleichen Ergebnisse zurückliefern würde. Näheres finden Sie im Abschnitt »Fensterfunktionen« auf Seite 243 in Kapitel 8.

## Text aus mehreren Feldern in einem einzigen Feld verketten

Dieser Abschnitt behandelt zwei unterschiedliche Szenarien:

- Text aus Feldern *in einer einzigen Zeile* in einem einzigen Wert verketten
- Text aus Feldern *in mehreren Zeilen* in einem einzigen Wert verketten

### Text aus Feldern in einer einzigen Zeile verketten

Die folgende Tabelle enthält zwei Spalten, und Sie wollen sie in einer einzigen Spalte verketten.

```
+----+---------+        +-----------+
| id | name    |        | id_name   |
+----+---------+        +-----------+
|  1 | Boots   |  --->  | 1_Boots   |
|  2 | Pumpkin |        | 2_Pumpkin |
|  3 | Tiger   |        | 3_Tiger   |
+----+---------+        +-----------+
```

Verwenden Sie die CONCAT-Funktion oder den Verkettungsoperator (||), um die Werte zusammenzubringen:

```
-- MySQL, PostgreSQL und SQL Server
SELECT CONCAT(id, '_', name) AS id_name
FROM my_table;

-- Oracle, PostgreSQL und SQLite
SELECT id || '_' || name AS id_name
FROM my_table;
```

```
+-----------+
| id_name   |
+-----------+
| 1_Boots   |
| 2_Pumpkin |
| 3_Tiger   |
+-----------+
```

### Potenzielle Erweiterungen

Kapitel 7, behandelt weitere Möglichkeiten zum Arbeiten mit String-Werten zusätzlich zu `CONCAT`, darunter:

- Ermitteln der Länge eines Strings
- Suchen von Wörtern in einem String
- Extrahieren von Text aus einem String

## Text aus Feldern in mehreren Zeilen verketten

Die folgende Tabelle listet auf, wie viele Kalorien jede Person verbraucht hat. Sie wollen die Kalorien jeder Person in einer einzigen Zeile verketten.

```
+------+----------+      +------+----------+
| name | calories |      | name | calories |
+------+----------+      +------+----------+
| ally |       80 | ---> | ally | 80,75,90 |
| ally |       75 |      | jess | 100,92   |
| ally |       90 |      +------+----------+
| jess |      100 |
| jess |       92 |
+------+----------+
```

Verwenden Sie eine Funktion wie `GROUP_CONCAT`, `LISTAGG`, `ARRAY_AGG` oder `STRING_AGG`, um die Liste anzulegen.

```
SELECT name,
       GROUP_CONCAT(calories) AS calories_list
FROM workouts
GROUP BY name;

+------+---------------+
| name | calories_list |
+------+---------------+
```

```
| ally | 80,75,90      |
| jess | 100,92        |
+------+---------------+
```

Dieser Code funktioniert in *MySQL* und *SQLite*. Ersetzen Sie in den anderen RDBMS `GROUP_CONCAT(calories)` durch Folgendes:

*Oracle*
```
LISTAGG(calories, ',')
```

*PostgreSQL*
```
ARRAY_AGG(calories)
```

*SQL Server*
```
STRING_AGG(calories, ',')
```

### Potenzielle Erweiterungen

Der Abschnitt »Zeilen in einem einzigen Wert oder einer einzigen Liste zusammenfassen« in Kapitel 8 bietet nähere Informationen dazu, wie Sie andere Trennzeichen neben dem Komma (`,`) benutzen, wie Sie die Werte sortieren und wie Sie eindeutige Werte zurückliefern können.

## Alle Tabellen finden, die einen bestimmten Spaltennamen enthalten

Stellen Sie sich vor, Sie hätten eine Datenbank mit vielen Tabellen. Sie wollen schnell alle Tabellen finden, die einen Spaltennamen mit dem Wort `city` darin enthalten.

### Lösung

In den meisten RDBMS gibt es eine besondere Tabelle, die alle Tabellen- und Spaltennamen enthält. Tabelle 10-1 zeigt, wie Sie in den einzelnen RDBMS diese Tabelle abfragen.

Die letzte Zeile jedes Codeblocks ist optional. Sie können sie einfügen, wenn Sie die Ergebnisse auf eine bestimmte Datenbank oder einen Benutzer eingrenzen wollen. Wenn Sie diese Zeile weglassen, werden alle Tabellen zurückgeliefert.

*Tabelle 10-1: Alle Tabellen finden, die einen bestimmten Spaltennamen enthalten*

| RDBMS | Code |
|---|---|
| MySQL | `SELECT table_name, column_name`<br>`FROM information_schema.columns`<br>`WHERE column_name LIKE '%city%'`<br>`  AND table_schema = 'my_db_name';` |
| Oracle | `SELECT table_name, column_name`<br>`FROM all_tab_columns`<br>`WHERE column_name LIKE '%CITY%'`<br>`  AND owner = 'MY_USER_NAME';` |
| PostgreSQL, SQL Server | `SELECT table_name, column_name`<br>`FROM information_schema.columns`<br>`WHERE column_name LIKE '%city%'`<br>`  AND table_catalog = 'my_db_name';` |

Die Ausgabe zeigt alle Spaltennamen an, die den Begriff `city` enthalten, und gibt auch die Tabellen an, in denen sie sich befinden:

```
+------------+-------------+
| TABLE_NAME | COLUMN_NAME |
+------------+-------------+
| customers  | city        |
| employees  | city        |
| locations  | metro_city  |
+------------+-------------+
```

*SQLite* besitzt keine Tabelle, die alle Spaltennamen enthält. Stattdessen können Sie manuell alle Tabellen anzeigen lassen und dann die Spaltennamen in jeder Tabelle zeigen:

```
.tables
pragma table_info(my_table);
```

### Potenzielle Erweiterungen

Kapitel 5, behandelt weitere Möglichkeiten, mit Datenbanken und Tabellen zu interagieren, einschließlich:

- Betrachten existierender Datenbanken

- Betrachten existierender Tabellen
- Betrachten der Spalten einer Tabelle

Kapitel 7, betrachtet weitere Möglichkeiten neben `LIKE`, nach Text zu suchen, darunter:

- = für die Suche nach einem exakten Treffer
- `IN` für die Suche nach mehreren Begriffen
- Regular Expressions für die Suche nach einem Muster

# Eine Tabelle aktualisieren, deren ID einer anderen Tabelle entspricht

Stellen Sie sich vor, Sie hätten zwei Tabellen: `products` und `deals`. Sie wollen die Namen in der `deals`-Tabelle mit den Namen der Objekte in der `products`-Tabelle aktualisieren, die eine passende `id` haben.

```
SELECT * FROM products;

+------+--------------------+
| id   | name               |
+------+--------------------+
|  101 | Mac and cheese mix |
|  102 | MIDI keyboard      |
|  103 | Mother's day card  |
+------+--------------------+

SELECT * FROM deals;

+------+--------------+
| id   | name         |
+------+--------------+
|  102 | Tech gift    | --> MIDI keyboard
|  103 | Holiday card | --> Mother's day card
+------+--------------+
```

## Lösung

Nutzen Sie eine `UPDATE`-Anweisung, um Werte in einer Tabelle mittels der `UPDATE ... SET ...`-Syntax zu modifizieren. Tabelle 10-2 zeigt, wie Sie dies in den einzelnen RDBMS erledigen.